НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Програмна система прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання»

Виконав:
студент IV курсу, групи КП-51
Гончар Максим Іванович _____

Керівник:
Старший викладач кафедри ПЗКС,
Гадиняк Р.А. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.т.н., доцент,
Дідковська М.В. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____ 2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Гончару Максиму Івановичу

1. Тема проекту «Програмна система прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання», керівник проекту Гадиняк Руслан Анатолійович, старший викладач, затверджені наказом по університету від «22» травня 2019 р. №1331-С.

2. Термін подання студентом проекту «__» червня 2019 р.

3. Вихідні дані для дипломного проектування: див. Технічне завдання.

4. Зміст проекту:
   - аналіз існуючого програмного забезпечення, що використовує настрої соціальних медіа для передбачення фінансових даних криптовалют;
   - формування вимог до способу збору повідомлень соціальних медіа, способу збору необроблених фінансових даних криптовалют, методів прогнозування курсу криптовалют;
   - формування вимог до відображення прогнозувань у користувацькому інтерфейсі веб-додатку;
   - порівняння існуючих інструментів розробки програмного забезпечення.
   - програмна реалізація алгоритму аналізу тексту новин соціальних медіа та алгоритму створення прогнозів;
   - програмна реалізація користувацького інтерфейсу веб-додатку для відображення отриманих передбачень;
   - тестування програмного алгоритму та користувацького інтерфейсу.

5. Перелік обов'язкового ілюстративного матеріалу:
   - функціональність програмних засобів (креслення);
   - компоненти та інтерфейси веб-додатку (креслення);

- структура бази даних (креслення);
- схема взаємодії програмних модулів (плакат);
- алгоритм аналізу повідомлень соціальних медіа (плакат);
- алгоритм прогнозування фінансових даних (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС, к.т.н.		

7. Дата видачі завдання «31» жовтня 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	16.11.2018	
2.	Розроблення та узгодження технічного завдання	09.12.2018	
3.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
4.	Розроблення алгоритму для прогнозування курсу криптовалют	16.01.2019	
5.	Підготовка матеріалів другого розділу дипломного проекту	10.02.2019	
6.	Програмна реалізація та тестування програмної системи	20.02.2019	
7.	Підготовка матеріалів третього розділу дипломного проекту	10.03.2019	
8.	Підготовка матеріалів четвертого розділу дипломного проекту	11.04.2019	
9.	Підготовка графічної частини дипломного проекту	19.05.2019	
10.	Оформлення документації дипломного проекту	26.05.2019	

Студент _____ Гончар М.І.

Керівник проекту _____ Гадиняк Р.А.

АНОТАЦІЯ

Даний дипломний проект присвячений створенню програмної системи прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання.

Проведено докладний аналіз наявних аналогів, що створюють передбачення курсу цифрових валют. Було проаналізовано необхідні функціональні та нефункціональні вимоги до розроблюваного продукту, виділено необхідні можливості, що має надавати система своїм користувачам.

Розроблена програмна система є веб-додатком, що містить динамічно оновлювані сторінки, призначені для відтворення результатів аналізу тональності текстів новин, показу прогнозу курсу криптовалют, а також наведення проміжних результатів аналізу текстів. У системі передбачена реєстрація нових користувачів адміністратором системи, а також обмеження доступу до певних можливостей веб-додатку неавтентифікованого користувача. У серверній частині програмного забезпечення розроблено та протестовано різні підходи до вирішення задачі аналізу тональності тексту, створено власний статистичний алгоритм для дослідження кореляцій між тональністю новин у соціальних медіа та курсом криптовалют Bitcoin, Ether та Litecoin. Створено відкритий API для доступу до результатів роботи системи сторонніми розробниками.

У даному дипломному проекті розроблено: модуль збору фінансових даних, модуль збору текстових даних, модуль аналізу тональності текстових даних, відкритий API для сторонніх розробників, модуль роботи із основною базою даних, модуль передбачення цін криптовалют, реалізовано графічні елементи та адаптивний дизайн web-сторінок, а також протестовано і збережено основні алгоритми машинного навчання.

ABSTRACT

The diploma project is devoted to the creation of a software system for predicting cryptocurrencies price by analyzing social media using machine learning algorithms.

A detailed analysis of the existing analogues that create predictions of the price of digital currencies was carried out. Thanks to the found systems, the necessary functional and non-functional requirements for the software product were analyzed, and the necessary functionality of the system was identified.

The implemented software system is a web application that contains dynamically updated web pages, designed to represent the results of the analysis of the news sentiment, visualize the forecast of the cryptocurrencies price, as well as some of the intermediate results. The system provides the opportunity to create new users within the system by system administrator, as well as restricting access to certain features of the web application to the unauthenticated user. The back end of the application was developed and tested using various approaches, there was also created own statistical algorithm to study the correlations between the sentiment of news in social media and cryptocurrencies price such as Bitcoin, Ether and Litecoin. An extensive API for third-party developers was implemented to represent results of system's work.

In this diploma project, modules for financial and text data scraping were developed, as well as a module for analyzing text sentiment, free to use API, a module to work with the main database, a module to forecast cryptocurrencies price, implemented adaptive web pages designs, as well as the main algorithms were tested and saved for later use.

ДП.045440-01-90 Програмна система прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання.
Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Програмна система	6	
	прогнозування курсу		
	криптовалют за		
	допомогою аналізу		
	соціальних медіа		
	алгоритмами машинного		
	навчання. Технічне		
	завдання		
ДП.045440-03-81	Програмна система	64	
	прогнозування курсу		
	криптовалют за		
	допомогою аналізу		
	соціальних медіа		
	алгоритмами машинного		
	навчання. Пояснювальна		
	записка		
ДП.045440-04-51	Програмна система	4	
	прогнозування курсу		
	криптовалют за		
	допомогою аналізу		
	соціальних медіа		
	алгоритмами машинного		
	навчання. Програма та		
	методика тестування		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-05-34	Програмна система	8	
	прогнозування курсу		
	криптовалют за		
	допомогою аналізу		
	соціальних медіа		
	алгоритмами машинного		
	навчання. Керівництво		
	користувача		
ДП.045440-06-99	Програмна система	1	
	прогнозування курсу		
	криптовалют за		
	допомогою аналізу		
	соціальних медіа		
	алгоритмами машинного		
	навчання.		
	Функціональність		
	програмних засобів. UML		
	діаграма прецедентів		
ДП.045440-07-99	Програмна система	1	
	прогнозування курсу		
	криптовалют за		
	допомогою аналізу		
	соціальних медіа		
	алгоритмами машинного		
	навчання. Компоненти та		
	інтерфейси веб-додатку.		
	UML діаграма		
	компонентів		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-08-99	Програмна система	1	
	прогнозування курсу		
	криптовалют за		
	допомогою аналізу		
	соціальних медіа		
	алгоритмами машинного		
	навчання. Структура		
	бази даних. ER діаграма		
ДП.045440-09-98	Програмна система	1	
	прогнозування курсу		
	криптовалют за		
	допомогою аналізу		
	соціальних медіа		
	алгоритмами машинного		
	навчання. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ПРОГРАМНА СИСТЕМА ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ ЗА ДОПОМОГОЮ АНАЛІЗУ СОЦІАЛЬНИХ МЕДІА АЛГОРИТМАМИ МАШИННОГО НАВЧАННЯ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ М.І. Гончар

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: програмна система прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості допоміжного інструменту аналізу курсу криптовалют інвесторами та трейдерами з метою зменшити ризики втрати інвестицій у певні цифрові валюти.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмна система повинна забезпечувати такі основні функції:

1. Можливість перегляду погодинного оновлюваних фінансових показників відслідковуваних криптовалют.
2. Щогодинний аналіз відслідковуваних сайтів новин, акаунтів соціальних мереж та інших інформаційних ресурсів.
3. Передбачення курсу криптовалют на основі зібраних фінансових даних та результатів аналізу даних соціальних мереж.

4. Можливість розширення структури системи для підтримки нових криптовалют.
5. Вивід та збереження результатів передбачення у інформативних графіках, діаграмах та таблицях.
6. Підтримка збору даних передбачень іншими розробниками.

Розроблення серверної частини виконати на мові програмування, використовуючи веб-фреймворк. Розроблення алгоритму передбачення виконати, використовуючи фреймворк машинного навчання. Розроблення клієнтської частини виконати, використовуючи бібліотеки для забезпечення інтерактивності та динамічності веб-сторінки.

Додаткові вимоги:

1. Отримання результатів передбачення, використовуючи або лише аналіз фінансових даних, або лише проаналізовані дані новин та соціальних мереж.
2. Підтримка змін джерел соціальних мереж для коригування аналізу передбачень.
3. Наявність адаптивного дизайну для двох платформ: настільних комп’ютерів та мобільних пристроїв.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1. Пояснювальна записка.
2. Програма та методика тестування.
3. Керівництво користувача.
4. Креслення:
   – «Функціональність програмних засобів. UML діаграма прецедентів»;

– «Компоненти та інтерфейси веб-додатку. UML діаграма компонентів»;
– «Структура бази даних. ER діаграма».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою проекту 16.11.2018
Розроблення та узгодження технічного завдання..... 09.12.2018
Підготовка матеріалів першого розділу дипломного проекту 30.12.2018
Розроблення алгоритму прогнозування курсу криптовалют..... 16.01.2019
Підготовка матеріалів другого розділу дипломного проекту 10.02.2019
Програмна реалізація та тестування програмної системи 20.02.2019
Підготовка третього розділу дипломного проекту..... 10.03.2019
Підготовка четвертого розділу дипломного проекту 11.04.2019
Підготовка графічної частини дипломного проекту 19.05.2019
Оформлення документації дипломного проекту 26.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНА СИСТЕМА ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ ЗА ДОПОМОГОЮ АНАЛІЗУ СОЦІАЛЬНИХ МЕДІА АЛГОРИТМАМИ МАШИННОГО НАВЧАННЯ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ М.І. Гончар

2019

ЗМІСТ

ВСТУП

Сучасні інвестиційні хедж-фонди та самостійні трейдери щодня проводять торги на різноманітних криптовалютних біржах, заробляючи на високій волатильності цифрових активів. Для того, щоб мати стабільні позитивні результати у вигляді прибутку при купівлі-продажу криптовалютних одиниць, інвесторам та трейдерам необхідно мати чітке та обґрунтоване бачення ціни віртуальної валюти через певний період часу в майбутньому. Саме тому прогнозування курсу цифрових валют є необхідною та однією з важливих діяльностей фахівців у галузі криптовалютного трейдингу.

Ринок криптовалют на даний момент має унікальну властивість – існування сильної залежності курсу від новин у популярних соціальних мережах, на веб-сайтах новин тощо. Саме тому одним із підходів для створення передбачень криптовалютних котирувань є машинний аналіз повідомлень від різних інформаційних джерел в інтернеті: центральних сайтів новин на кшталт Cointelegraph та CCN, публічних груп у соціальних мережах Facebook, Twitter та Reddit, форуму Bitcointalk тощо. Аналіз таких ресурсів складається з автоматизованого визначення узагальненої позитивної, негативної чи нейтральної позиції в останніх текстових повідомленнях щодо певного активу, на основі чого можна зробити висновок про майбутній тренд: збільшення чи зменшення ціни певної криптовалюти.

Створення системи, що впроваджує такий спосіб прогнозування цін цифрових валют, є актуальною задачею, оскільки криптовалютний трейдинг продовжує набувати своєї популярності завдяки зростальному інтересу інвесторів до віртуальних валют. Актуальність задачі також підкріплюється тим, що схожий підхід до створення передбачень цін можна застосовувати не тільки до криптовалют, а й до традиційних активів, оскільки показник

ставлення медіа до вибраного активу є одним із впливових чинників зростання чи спадання її ціни.

Даний дипломний проект присвячено розробці системи у вигляді веб-додатку, призначеної для збору текстових даних різноманітних онлайн-медіа, їх аналізу та побудови прогнозу курсу криптовалют, використовуючи алгоритми машинного навчання.

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Трейдер – учасник торгівлі на біржі, що заробляє на торговельних операціях продажу-купівлі певних цінних активів (акцій, валюти тощо).

Ліквідність – у економіці, властивість цінних активів бути швидко проданими за ціною, близькою до ринкової. Чим легше та швидше можна обміняти актив, враховуючи його повну вартість, тим більш ліквідним він є.

Криптовалюта – цифрова, віртуальна валюта, що використовує криптографічні алгоритми асиметричного шифрування для забезпечення безпеки транзакцій. Криптовалюту також називають криптомонетою.

Асиметричне шифрування – спосіб шифрування даних, у якому використовується пара пов’язаних ключів: публічного та приватного.

Криптовалютний трейдинг – один із видів інтернет-трейдингу, торгівля на криптовалютній біржі.

Forex, FOReign Exchange – міжбанківський міжнародний валютний ринок.

Bitcoin, біткоін – один із видів цифрової валюти, криптовалюта. Bitcoin може бути проданий в обмін на товари або послуги, що приймають Bitcoin у якості платежу.

Альткоін – будь-яка цифрова криптовалюта, відмінна від Bitcoin. Термін походить від «альтернатива до Bitcoin» та використовується для описання криптовалюти, що не є Bitcoin.

Фіатні гроші – тип грошей або валюти, цінність яких походить не від власної вартості або гарантії обміну на золото або іншу валюту, а від державного наказу використання їх як засобу платежу.

Графік «японські свічки» – вид інтервального графіка і технічний індикатор, що застосовується для показу змін біржових котирувань акцій, цін на сировину тощо.

RSI, Relative Strength Index, індекс відносної сили – індикатор технічного аналізу, що визначає силу тренду та ймовірність його зміни в майбутньому.

ICO, Initial Coin Offering – одна із форм колективного фінансування, краудфандингу, спосіб залучення фінансових інвестицій у проект. При такому способі збору інвестицій випускається валюта, яка розподіляється серед зацікавлених осіб.

Forex сигнал – порада від стороннього сторони щодо початку торгівлі певного активу за певною ціною та в певний час.

Freemium модель монетизації – бізнес-модель, що полягає в наданні користувачу можливості скористатись базовою безкоштовною або платною і покращеною версією продукту.

API, Application Programming Interface – у програмному забезпеченні, набір використовуваних функцій, інтерфейс для створення програмних додатків.

RESTful API – API, що використовує HTTP запити GET, PUT, POST та DELETE для забезпечення доступу до інформації всередині системи стороннім користувачам.

Back end – частина комп'ютерної системи або програми, до якої безпосередньо не звертається користувач. Зазвичай відповідає за зберігання та маніпулювання даними.

Front end – інтерфейс для взаємодії між користувачем і back end.

Десктопна програма – програма, призначена для використання на настільних комп'ютерах.

Chrome V8 engine – програма, що оброблює скрипти JavaScript у браузері.

Couroutines, корутини – структура управління виконання коду, схожа на потоки, що має свій власний стек, свої власні локальні змінні та дозволяє виконувати певною мірою паралельний код. У мові Python корутини створюються за допомогою ключового слова yield.

Callback, колбек – у Node.js, асинхронний еквівалент функції, що викликається по закінченні певної задачі.

DOM, Document Object Model – інтерфейс, що дозволяє програмам та скриптам динамічно отримувати доступ і оновлювати вміст, структуру та стиль html документа.

Ajax, Asynchronous JavaScript and XML – технологія для побудови інтерактивних веб-додатків, яка оброблює запити користувача одразу при їх приході.

jQuery – бібліотека JavaScript, що дозволяє веб-розробникам додавати функціональність до веб-сторінки, наприклад: обробити дані форми, модифікувати текст, перемістити елемент на сторінці тощо.

SVG, Scalable Vector Graphics – формат зображень на основі XML для зображення двовимірної графіки, що підтримує інтерактивність та анімації.

NLP, Natural Language Processing – автоматизація маніпуляцій природної мови програмним забезпеченням, наприклад, розмови або тексту.

N-грам – комбінація суміжних слів або літер довжиною N, які можна знайти у вихідному тексті.

ER діаграма – графічна ілюстрація сутностей інформаційної системи та взаємовідносин між цими сутностями.

СКБД, Система Керування Базою Даних – інструментарій для управління базою даних та маніпуляції даних всередині неї.

USD, United States Dollar – долар США.

JSON, JavaScript Object Notation – один із форматів текстового обміну даними, є легко зчитуємим для людини, використовується для представленя структур даних (масив, словник тощо) у текстовому вигляді.

Footer – елемент HTML-документа, що знаходиться у нижній частині веб-сторінки.

CDN, Content Delivery Network – розподілена система, що зберігає кешовані корисні файли для надання доступу користувачам по всьому світу.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1. Аналіз особливостей криптовалютного трейдингу

Інтернет-трейдингом називають торгівлю на фінансових фондових ринках за допомогою онлайн сервісів бірж Binance, OKEx, DOBI Exchange. Перевагами такого способу купівлі-продажу активів є низька вартість операцій для трейдерів, можливість заробити на підвищеній ліквідності, впровадження автоматизованої машинної торгівлі, що може працювати за власним алгоритмом в автоматичному режимі тощо [1, 2].

Криптовалютою називають таку цифрову валюту, що функціонує завдяки механізму асиметричного шифрування. На даний момент існує більше ніж дві тисячі таких валют, що доступні для торгівлі, найвідомішими з яких є Bitcoin, Ether, Ripple, Bitcoin Cash, Litecoin [3, 4]. Станом на квітень 2019 року загальна капіталізація складає 176 млрд доларів США, середній обсяг торгівлі за 24 години – 50 млрд доларів США [4]. Попит на криптовалюти обумовлений інвестиційним інтересом, можливістю вигідного переказу валюти по світу та спекулятивним чинником.

Хоча у більшості країн світу криптовалюти ще не набули правового статусу, дані цифрові активи широко застосовуються для проведення трейдингових спекулятивних операцій. Криптотрейдинг, або криптовалютний трейдинг – це процес торгівлі криптовалютами на криптовалютних біржах. Трейдер має можливість як купити та продати одиниці валюти за фіатні гроші, так і обміняти одну цифрову валюту на іншу. Наприклад, використовуючи такі криптовалютні біржі як Bitfinex, BitStamp чи Coinbase, трейдер має можливість купити за долар США криптовалюту Bitcoin, Ether, Ripple, Litecoin чи інший альткоін, з яким можна проводити подальші операції обміну [1, 2].

Технічний аналіз – метод оцінки фінансового ринку, завдяки якому можна робити більш виважені та математично і статистично обґрунтовані передбачення щодо поведінки ціни певного активу в майбутньому [5].

Використовуючи показники такого аналізу, трейдери мають можливість чітко виділити і передбачити важливі висхідні, низхідні та нейтральні тренди ринку. За допомогою прогнозування на основі технічного аналізу існує можливість знайти нові прибуткові стратегії торгівлі. Застосування технічного аналізу є вигідним для інвестора для передбачення курсу криптовалют, оскільки такий підхід значно зменшує ризик прийнятих рішень купівлі або продажу високоволатильних активів [1, 5, 6].

Популярним способом технічного аналізу є аналіз графіків японських свічок, де кожна свічка показує на рух ціни цифрового активу за певний інтервал часу. Цінність аналізу графіків японських свічок є виділення різноманітних закономірностей руху курсу, що дає можливість трейдеру передбачити поведінку криптовалютного ринку [5]. Ще одним технічним показником є RSI. Завдяки цьому індикатору є можливість побачити силу тренду, оцінити реальну цінність криптовалюти, визначити, чи є валюта у перекупленому чи не зовсім докупленому стані [5].

Технічний аналіз сам по собі може не дати високої точності при передбаченні фінансових даних. Одним зі способів уточнення прогнозів є комбінування методів технічного аналізу з аналізом тональності повідомлень соціальних медіа [7, 8, 9]. Даний метод передбачає створення моделі, яка оцінює курс криптовалюти, виходячи із таких показників: кількість новин чи коментарів на тему, аналіз тональності тексту новин, технічний аналіз історичних цін та інших.

Процес аналізу текстів новин може бути виконаний статистичними методами [7, 8, 9], методами, що використовують елементи машинного навчання [10, 11, 12] та гібридними методами [13, 14]. Такі моделі вирішують задачі знаходження ставлення автора в тексті до певного факту, виявлення певних характеристик знайденої оцінки в залежності від поставленої задачі. Метою створення такої моделі є визначення гіпотези, що визначає той факт, чи є текст позитивно, негативно чи нейтрально налаштованим до конкретного об’єкта. Результатом аналізу тональності

тексту є визначена оцінка тональності – позитивна, негативна, нейтральна. Прикладом позитивної тональності є текст «Капіталізація компанії зростає другий рік підряд», прикладом негативної тональності є «Акції Apple обвалились на фоні торговельної війни». Під нейтральною оцінкою мається на увазі той факт, що текст не містить емоційного забарвлення стосовно певної теми або те, що алгоритм не знайшов згадок шуканої тематики. Основними труднощами такого способу аналізу тексту є наявність сленгу, сарказму автора, фразеологізмів, абревіатур, специфічних термінів на кшталт «бичачий тренд» тощо у досліджуваних реченнях – такі фактори можуть ввести в оману не тільки людей, а й комп'ютерні алгоритми. Одними зі способів, що допомагають подолати такі труднощі, є реалізація лексичних класифікаторів та застосування специфічних алгоритмів машинного навчання [10, 11, 12, 13].

Алгоритмами машинного навчання називають такі алгоритми, що мають можливість навчатись при наявності коректних даних та покращувати свою продуктивність роботи із часом, без втручання людини. Том Мічел (Tom Michael Mitchell), професор американського університету Карнегі-Мелон (Carnegie Mellon University), визначає термін машинного навчання так: «Комп'ютерна програма отримує досвід E виконання класу задач T із показником продуктивності P, якщо продуктивність P виконання задач класу T покращується із досвідом E» [15]. Основними типами машинного навчання є класичне навчання (supervised learning, unsupervised learning), навчання із підкріпленням (reinforcement learning) та нейронні мережі (neural networks) [16, 17]. Нейронні мережі сьогодні використовують для таких задач, як визначення об'єктів на фото та відео, розпізнавання та синтез мови, обробки фото, машинного перекладу. Основна задача класичного машинного навчання з учителем – виявити залежність у формі гіпотези між відомими входами та еталонними виходами [17]. Алгоритми, що використовують елементи машинного навчання, можуть також вирішувати задачу регресії, таким чином

передбачаючи дані змінної неперервної величини, наприклад, курсу криптовалют, на основі знань у минулому.

Моделі, побудовані на алгоритмах машинного навчання, сьогодні широко використовуються трейдерами для автоматизованого прогнозування курсу як традиційних активів, так і криптовалют, оскільки часто дозволяють зробити більш точні передбачення, ніж статистичні методи чи прогнози, побудовані на досвіді живих професійних трейдерів. Не зважаючи на те, що в сучасних фінансових ринках існує велика кількість різноманітних факторів, що впливають на курс активу, таким алгоритмам вдається із певним високим відсотком точності передбачити майбутні тренди: дізнатись напрямок ходу ціни на тривалий або короткочасний термін, оцінити тривалість «бичачого» (направленого на підвищення цін у майбутньому), чи «ведмежого» (направленого на зниження цін у майбутньому) настрою гравців на біржі [18, 19]. Ще однією важливою властивістю алгоритмів машинного навчання є можливість визначити невідомі характеристики (features), які можна використовувати при створенні передбачення котирування криптовалюти [19].

1.2. Огляд існуючих сервісів із прогнозування курсу криптовалют

При проведенні пошуку наявних рішень розроблюваного ПЗ було визначено певні відкриті рішення у вигляді веб-сервісів, що надають аналіз цін криптовалют, використовуючи аналіз повідомлень із соціальних мереж. Усі зазначені проекти мають підтримку спільноти в інтернеті та показують активність у своїх соціальних мережах, що свідчить про їх адекватні результати та зацікавленість користувачів у результатах подібних сервісів.

Веб-сервіс coinpredictor.io [20] є одним зі знайдених наявних аналогів. Основна функціональність системи складається з надання передбачень котирувань криптовалют після аналізу наступних показників: ціна віртуальної валюти, кількість новин, пов’язаних із вибраною криптомонетою, аналіз та передбачення цін монет ICO, значення кореляції

між криптовалютами та їх вплив одна на одну, аналіз тональності новин зі згадкою певної цифрової валюти. Сервіс також надає календар із майбутніми масштабними подіями, пов'язаними із певною криптовалютою (конференція, випуск нового протоколу тощо) та інформацію про вплив таких подій на майбутній курс. Модель монетизації даного сервісу – рекламні інтеграції на сайті та продаж приватних передбачень від розробників системи. Передбачення представлені у вигляді графіків та таблиць із зазначенням позитивного чи негативного тренду щодо певного активу.

Приклад користувацького інтерфейсу для отримання передбачення у веб-сервісі coinpredictor.io зображений на знімку екрана на рис. 1.1. На цьому рисунку зображене передбачення середньої ціни цифрових валют Bitcoin, Litecoin та Binance Coin у форматі інтерактивного графіка, показана кількість проведених публічних подій за минулий рік та кількість майбутніх важливих публічних подій у поточному році.

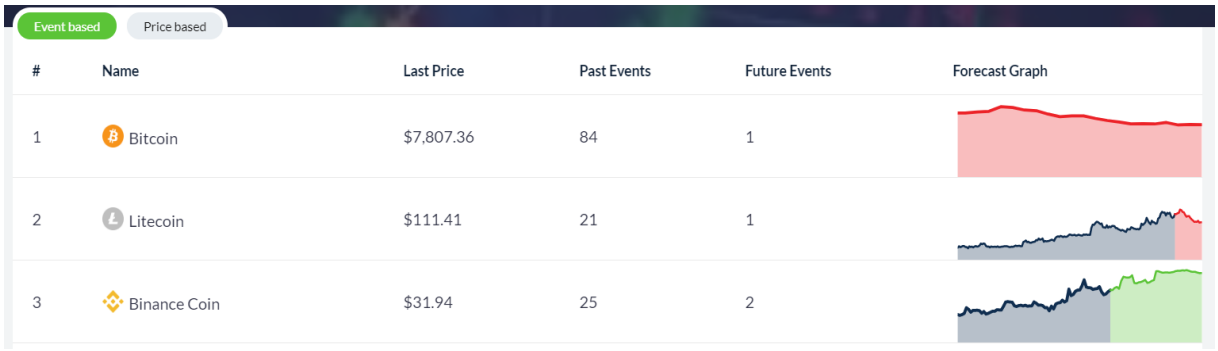


Рис. 1.1. Передбачення криптовалют Bitcoin, Litecoin та Binance Coin у веб-сервісі coinpredictor.io

Веб-сервіс також надає прогнози фінансового стану ICO проектів, аналізуючи результати краудфандингових кампаній та зіставляючи метрики із вже існуючими успішними проектами. Приклад користувацького інтерфейсу для отримання прогнозів цін ICO зображений на знімку екрана на рис. 1.2. На цьому рисунку зображені дати початку та кінця проведення

кампанії проекту «Emiba» та очікуване передбачення зросту ціни випущеної монети на кінець завершення кампанії.

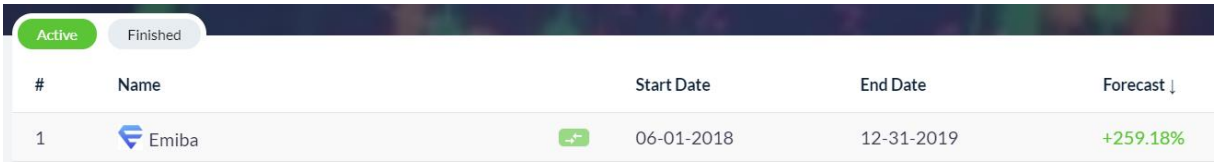


Рис. 1.2. Передбачення ICO Emiba у веб-сервісі coinpredictor.io

Наступною знайденою програмною розробкою є веб-сервіс santiment, розташований за адресою santiment.net [21]. Даний проект забезпечує користувача інформацією про сигнали у реальному часі, схожі на Forex сигнали, та про тренди криптовалют на основі більше ніж 50 метрик тональності з інтернет-ресурсів новин.

Основною розробкою даного сервісу є частина системи, названа «SANbase» – веб-сторінка, на якій зображена інформація про майбутні тренди та факти, що свідчать про підвищений інтерес до певного цифрового активу. Наступною розробкою сервісу sentiment є «SANgraphs», що створює графіки основних метрик криптовалюти, описуючи закономірності, пояснюючи напрямок ціни та візуалізуючи інші показники технічного аналізу. У веб-сервісі santiment є також можливість завантажувати аналіз фінансових даних у вигляді таблиць за допомогою розробки «SANsheets».

Модель монетизації даного сервісу – freemium модель, у якій підписка на розширені плани надає доступ до користування розширеними та більш професіональними можливостями системи. Даний сервіс також надає веб-орієнтоване робоче місце та доступ до результатів своєї роботи розробниками через власний RESTful API.

На сайті надана інформація про ті ключові слова, що є найбільш вживаними за останній місяць у медіа, які пишуть новини про цифрові валюти. Приклад такої інформації у вигляді таблиці показаний на рис. 1.3.

На знімку екрана показані метрики «Hyper score» та «Social volume» – штучний показник популярності певного слова та кількість згадувань за останню добу.

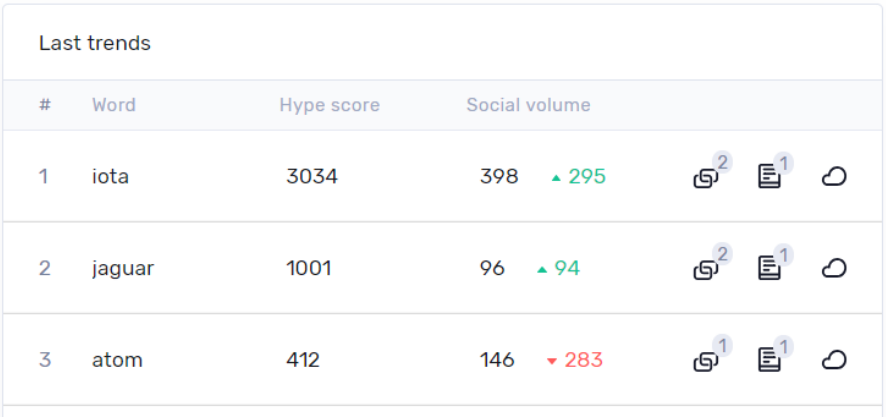


Рис. 1.3. Інформація про популярні ключові слова (iota, jaguar, atom) за останні 24 години у веб-сервісі santiment

Веб-сервіс надає користувачу можливість переглянути хмарку тегів (візуалізація зваженого списку найчастіше вживаних слів), завдяки чому можна побачити контекст використання ключового слова у повідомленнях медіа. Приклад такої хмарки наведений на рис. 1.4.

Рис. 1.4. Хмарка тегів у веб-сервісі santiment, що показує контекст згадування слова «iota» у новинах

Інструмент santiment дозволяє також переглянути кореляцію між курсом криптовалют та кількістю згадувань у соціальних мережах та новинах. Приклад даних про курс криптовалюти Bitcoin зображений у вигляді графіка, показаний на рис. 1.5. На цьому графіку чітко видно, що дані про кількість згадувань криптовалюти в інтернет-ресурсах мають пряму залежність від ціни криптовалюти на всьому розглянутому часовому проміжку. Наприклад, після початку підвищення кількості згадувань криптовалюти Bitcoin 30 березня відбувся різкий зріст котирувань криптовалюти.

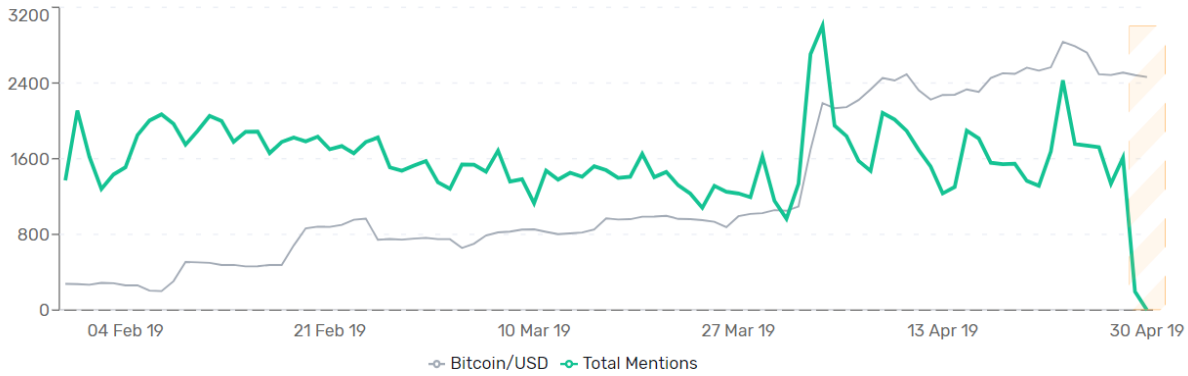


Рис. 1.5. Графік аналізу курсу криптовалют та його згадувань у інтернет-ресурсах у веб-сервісі santiment

Ще одним знайденим аналогом розроблюваного програмного забезпечення є веб-сервіс The TIE [22]. Дане розроблення надає трейдерам криптовалют повний набір інструментів, необхідних для ідентифікації сигналів, управління активами та проведення успішних торгів. На веб-сторінці The TIE можна переглянути загальний тренд ринку, тренди криптовалют із найбільшою капіталізацією, криптовалюти із найнижчим та найвищим параметром настрою у соціальних медіа, криптовалюти із найбільшою кореляцією однієї до одної. За допомогою The TIE можна переглянути кореляцію між ціною криптовалюти, кількістю твітів та оцінкою поточного настрою соціального медіа на одному графіку,

зображеному на рис. 1.6. Варто зазначити, що на цьому графіку виділені деякі новини, що мали найбільший вплив на зміну курс криптовалюти.

Рис. 1.6. Кореляція ціни Bitcoin із метриками настроїв соціальних медіа у веб-сервісі The TIE

Даний веб-сервіс використовує аналіз тональності повідомлень із Twitter – твітів – показуючи кількість новин про певну криптовалюту та загальне ставлення авторів твітів до криптовалюти за останні 24 години. Метрики настроїв зображені у вигляді «спідометра», зображеного на рис. 1.7, стрілка якого вказує на поточне ставлення авторів новин соціальних медіа до певної цифрової валюти.

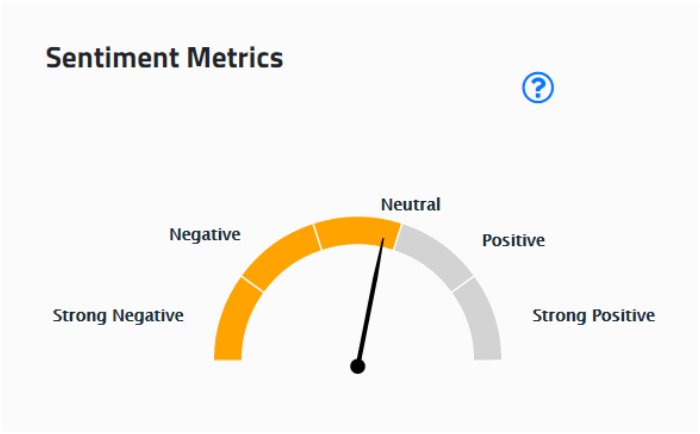


Рис. 1.7. Візуалізація поточної тональності соціальних новин до валюти у веб-сервісі The TIE

Варто зазначити, що зазначені вище веб-сервіси явно визначають у правилах свого використання той факт, що їх передбачення не є абсолютно точними, а представлення передбачень на сайті не є підставою для початку торгів. Чітко підкреслюється, що сервіси є лише допоміжними інструментами для кращого розуміння трендів криптовалютного ринку з точки зору соціальних медіа.

Поруч із готовими рішеннями було знайдено велику кількість наукових робіт, присвячених темі прогнозуванню як традиційних [23], так і криптовалютних активів [24] за допомогою аналізу інтернет-ресурсів новин, що свідчить про серйозний інтерес наукової спільноти до тематики аналізу контенту в онлайн джерелах та їх впливу на фінансові дані.

Передбачення цін активів на фондових біржах за допомогою аналізу повідомлень соціальних мереж також часто застосовується в приватних інвестиційних фондах. Через те, що ці фонди приватні, а їх програмне забезпечення для прогнозування цін є закритим, не вдалось знайти готові для використання моделі, що використовуються в хедж-фондах.

1.3. Аналіз вимог до функціональності програмних засобів

Проаналізувавши сферу криптовалютного трейдингу та наявні аналоги програмного забезпечення для прогнозування курсу криптовалют, можна виділити наступні функціональні вимоги для розроблюваного дипломного проекту:

1. Система має надавати прогноз ціни криптовалют Bitcoin, Ether та Litecoin на наступну годину, добу та місяць, використовуючи аналіз новин та повідомлень соціальних мереж.
2. Система має проводити збір поточної ціни криптовалют Bitcoin, Ether та Litecoin.
3. Система має збирати текстові дані новинних ресурсів та повідомлень соціальних медіа Twitter та Reddit, а також сайтів новин CoinTelegraph та CCN, що пишуть про досліджувані криптовалюти.

4. Система має фільтрувати та обробляти зібрані тексти новин таким чином, щоб відформатовані дані були прийнятними для використання алгоритмами машинного навчання для визначення тональності тексту.
5. Система має надавати оцінку тональності тексту новини чи повідомлення від соціальних медіа щодо криптовалюти Bitcoin, Ether чи Litecoin за допомогою алгоритмів машинного навчання.
6. Система має надавати узагальнене значення метрик настрою соціальних мереж та новин про криптовалютний ринок.
7. Аналіз тональності тексту має відбуватись у контексті криптовалют, трейдингу, торгівлі на біржі.
8. Система має показувати інформацію про тональність повідомлень у соціальних медіа про узагальнений криптовалютний ринок.
9. Система має показувати стрічку останніх зібраних новин про криптовалюти Bitcoin, Ether та Litecoin.
10. Система має надавати стороннім розробникам доступ до результатів передбачення через власний API.
11. Система має повідомляти користувача про те, що представлена інформація в сервісі опублікована виключно в освітніх цілях, а не для надання інвестиційних консультацій.
12. Програмна система має забезпечувати візуалізацію проведеного аналізу та передбачення за допомогою графіків, діаграм та таблиць, у яких зазначено результати роботи основних алгоритмів.
13. Система має надавати можливість ручного оновлення фінансових та текстових даних адміністратором системи.
14. Система має надавати можливість адміністратору реєструвати довірених осіб, що можуть вручну редагувати тональність текстів.

На основі функціональних вимог, зазначених вище, було складено діаграму прецедентів веб-додатку (див. Додаток 1).

Також було визначено нефункціональні вимоги до розроблюваної системи:

1. Збір фінансових даних має проходити кожну годину.
2. Збір текстових даних новин та повідомлень соціальних медіа має відбуватись кожну годину.
3. Візуалізація результатів має бути доступна користувачу після того, як користувач погодився із правилами користування системою.
4. Точність передбачення курсу криптовалют Bitcoin, Ether та Litecoin має бути не менше ніж 60%.
5. Дизайн веб-сторінки має бути коректно показаний у веб-браузерах Google Chrome, Mozilla Firefox, Safari, Microsoft Edge.
6. Дизайн веб-сторінки має бути коректно показаний як на мобільних пристроях, так і на настільних комп'ютерах.
7. Для збору фінансових даних криптовалют програмною системою має бути використаним RESTful API Cryptocompare та ICObench.
8. Для збору текстових даних новин система має використовувати ресурси Twitter, Reddit, CCN та CoinTelegraph.
9. У розроблюваному веб-додатку мають бути реалізовані ролі користувача, довіреної особи та адміністратора системи.
10. Алгоритм аналізу тексту має бути масштабованим: готовим до роботи із різноманітними текстовими ресурсами.
11. Система має бути готова до розширення кількості криптовалют та монет ICO, курс яких необхідно передбачити.
12. Візуалізація результатів роботи системи, а також інтерфейс користувача мають бути прості для сприймання.
13. Ручне оновлення текстових новин має тривати не більше 5 хвилин за розглянутий день, а ручне оновлення фінансових новин має тривати не більше ніж 2 хвилини за розглянутий день.

2. АНАЛІЗ МОВ ПРОГРАМУВАННЯ, ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ ВЕБ-ДОДАТКІВ ТА АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

2.1. Вибір веб-додатку в якості типу програмного забезпечення

Бізнес-ідею можна втілити в життя за допомогою програмування в різноманітних формах: як веб-застосунок, як додаток, орієнтований на Windows, Linux, MacOS, як мобільний додаток тощо. Програміст має знати, як саме користувачі будуть використовувати розробку та які інструменти необхідні для самої розробки – саме тому вибір способу коректної реалізації програмного забезпечення є важливим етапом у проектуванні.

Розглянемо два можливі типи додатку: десктопна програма та веб-додаток. Клієнтська та серверна частини можуть бути реалізовані в обох формах.

Веб-додатки мають низку переваг [25]:

1. Веб-програми не потребують значних зусиль при розгортанні на машині клієнта. Для доступу до такого типу програмного забезпечення необхідний лише веб-браузер.
2. Оновлення веб-додатків приходять частіше, однак користувачам не потрібно постійно локально оновлювати версію додатка. Також, завдяки швидким оновленням, програміст, що розроблює веб-додаток, може легше та швидше усувати помилки в програмі.
3. Незалежність платформи. Веб-розробник може не турбуватись про системні виклики чи бібліотеки, що відрізняються у різних операційних системах.
4. Можливість впровадження адаптивного дизайну на екранах із різною роздільною здатністю, у тому числі на мобільних пристроях.

5. Швидкість та ціна процесу розроблення веб-додатку значно менша, ніж десктопного додатку за рахунок меншого необхідного рівня кваліфікації розробників.

Веб-додатки мають також низку недоліків:

1. Залежність користувача від підключення до інтернету.
2. Низький рівень безпеки персональних даних користувача, оскільки всі дані зберігаються у віддаленій базі даних, що частіше за все знаходиться у незашифрованому хмарному сховищі.
3. Деякі браузери можуть не підтримувати технології, використані у веб-додатку.

У свою чергу, програми для настільних комп'ютерів мають свої переваги:

1. Програміст має можливість оптимізувати програмне забезпечення для конкретної операційної системи.
2. Десктопні програми мають можливість напряму з'єднуватись з обладнанням машини, наприклад, із відеокартою чи звуковою картою, через системні виклики певної операційної системи.
3. Інтерфейс операційної системи може бути значно зручнішим для користувача, ніж інтерфейс веб-браузера.
4. Процес розгортання застосунку користувачем легший: необхідно лише використати інсталятор програми, а не реєструвати сервер у мережі.
5. Користування десктопною програмою безкоштовне. Власник веб-додатку у свою чергу має заплатити за хостинг, сховище даних та ін.
6. Десктопна програма може працювати швидше, ніж веб-додаток, при поганому підключенню клієнта до інтернету.

Програмне забезпечення для персональних комп'ютерів також має деякі недоліки:

1. Доступ до додатка обмежений одним персональним комп'ютером.

2. Програма реалізує та використовує свій власний протокол для роботи, замість того, щоб користуватись існуючим, добре задокументованим та універсальним протоколом.

Важливо пам’ятати, що через деякий час може виникнути необхідність у масштабуванні програмного забезпечення як у сторону веб-додатків, так і у сторону десктопних програм, так і у сторону мобільних додатків.

Варто зазначити, що існуючі аналоги розроблюваного програмного забезпечення, описані у підрозділі 1.2, існують та підтримуються лише у формі веб-додатків.

Оцінивши функціональні та нефункціональні вимоги до програмної розробки у підрозділі 1.3, а також визначивши переваги та недоліки деяких типів програмного забезпечення, описаних вище, було вибрано реалізацію програмного застосунку у формі веб-додатку із розділенням на клієнтську частину (front end) та серверну частину (back end).

2.2. Мови програмування для back end частини ПЗ

На початку розроблення програмного проекту програміст має виконати дослідження та запланувати, які мови програмування і технології є доречними для розроблення певного продукту. Back end частина веб-додатку складається із коду, що обробляє дані запитів від клієнтської частини, опрацьовує запити до бази даних, виконує основну логіку програми.

Найпоширенішими мовами програмування для розроблення веб-додатків на початок 2019 року є JavaScript, PHP, Java та Python [26]. Усі мови мають широку підтримку розробників, на цих мовах написані випробувані часом фреймворки та бібліотеки для розроблення веб-додатків, а також існує безліч прикладів готового програмного забезпечення.

PHP та Java заслуговують на увагу у якості основної мови для написання серверної (back end) частини веб-додатку: використовуючи

фреймворки Spring, Hibernate чи Apache Struts можливо написати веб-додаток на мові Java, а із допомогою Symfony, Laravel чи Yii2 можна створити веб-застосунок на мові PHP.

Основну увагу варто приділити мовам програмування JavaScript та Python, оскільки саме їх популярність зростає з кожним роком серед розробників веб-додатків. На кінець 2018 року найуживанішими фреймворками для веб-розроблення додатків на мові Python були Flask та Django, а для розроблення на мові JavaScript – фреймворки, що були побудовані на основі Node.js: Express.JS, Socket.io та Meteor.JS [27, 28].

Одним із пунктом для порівняння цих мов є швидкодія програми: як швидко застосунок відповідає на дію користувача. Порівнюючи швидкодію веб-фреймворків Python із Node.js, можна помітити, що Node.js працює значно швидше. Такий високий рівень продуктивності значною мірою пояснюється тим, що Node.js базується на швидкому та потужному Chrome V8 engine. Саме тому додатки, що працюють у реальному часі, наприклад, онлайн-чати, краще писати на Node.js. Крім того, веб-фреймворки на мові Python не працюють оптимально в тих додатках, що вимагають великої кількості оперативної пам’яті. Це означає, що у тому випадку, коли веб-додатку треба буде працювати із високонавантаженою 3D-графікою, програмісту рекомендується використовувати Node.js у якості основного інструмента для розроблення back end частини застосунку.

Ще одним аспектом для порівняння веб-фреймворків Python із Node.js є масштабованість та паралельна робота веб-додатків. Масштабованість є здатністю програми задовольняти зростаючу кількість запитів, не поступаючись своєю швидкодією. Така властивість має серйозне значення у високонавантажених додатках, що, наприклад, обслуговують велику кількість користувачів як із мобільних додатків, так із настільних комп’ютерів. Node.js при використанні створює однопоточну асинхронну архітектуру, у якій операції вводу та виводу завершуються за межами

потоку, не блокуючи його. Така властивість гарантує плавну масштабованість Node.js у простих веб-додатках, однак розроблення складних додатків із безліччю паралельних процесів вимагає глибоких теоретичних знань, значної уваги розробника та попереднього ретельного інженерного дослідження. Веб-фреймворки, що написані на мові Python, у свою чергу, не підтримують асинхронне програмування за замовчуванням, але підтримують корутини (coroutines), за допомогою яких є можливість домогтись асинхронної обробки запитів. Таким чином, мова програмування Python має повний інструментарій для досягнення необхідної масштабованості веб-додатку.

Наступним пунктом для порівняння цих мов програмування в контексті розроблення веб-додатків є оброблення помилок. Простота та прозорість обробки помилок є критичним тезисом при виборі мови програмування. Як JavaScript, так і Python добре справляються із пійманням та обробленням помилок, що виникають під час виконання коду. Але у більшості випадків оброблення виняткових ситуацій у Python простіша та займає менше часу для налагодження та виправлення помилок.

Важливим аспектом при виборі мови програмування є простота написання коду та швидкість освоєння нових технологій програмістами. Мова програмування Python підтримує такий стиль програмування, що є визнаним серед розробників як найлегший для написання, підтримки та його подальшого сприймання.

Веб-фреймворк Django, написаний на Python, має високий рівень захисту сервера та його даних, відмінно працює із різноманітними базами даних, легко масштабується, має велику спільноту розробників та докладну документацію. Недоліками Django є його монолітність, тобто неможливість розбити додаток на декілька рівнів – мікросервісів, він не є зручним для написання великих програм, оскільки у такому разі потребує високого рівня теоретичних і практичних знань фреймворку.

Node.js, у свою чергу, може використовувати велику кількість бібліотек від спільноти, має високу продуктивність і швидкодію, відмінний для створення сервісів з API, легко обробляє одночасні запити та користується популярністю серед спільноти розробників програмного забезпечення. Основним недоліком Node.js є його неспроможність працювати з додатками, що використовують багато ресурсів процесора, оскільки Node.js працює в одному потоці. Ще одною вадою Node.js є те, що наявність колбеків (callbacks) у програмному вихідному коді часто призводить до виникнення великої вкладеності, що має поганий вплив на розуміння коду та його швидкодії.

2.3. Технології розроблення front end частини ПЗ

Front end веб-розроблення, також відома як розроблення на стороні клієнта – це практика створення HTML сторінок, CSS файлів і JavaScript скриптів для веб-сайту або веб-застосунку, для того, щоб користувач міг безпосередньо взаємодіяти із контентом веб-сайту [29]. Основною метою розроблення front end частини веб-додатку є забезпечення повної в обсязі та легкої для сприймання інформації при відкритті користувачем веб-сторінки у браузері. Ця задача ускладнена тим фактом, що користувачі використовують велику різноманітність пристроїв з різними розмірами та роздільною здатністю екрана, що змушує дизайнера взяти до уваги ці аспекти при проектуванні сайту. Користувачам також необхідно забезпечити умови, при яких їх сайт правильно з’являється в різних браузерах, на різних операційних системах та на різних пристроях, що вимагає ретельного планування на стороні розробника front end частини програмного забезпечення.

Основними технологіями для створення клієнтської частини веб-сторінки для показу результатів роботи системи користувачу є HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) та мова програмування JavaScript.

HTML – мова розмітки документів, що використовується для створення веб-сайтів. За допомогою цієї мови розмітки розробник може показати вміст веб-сторінки у вигляді блоків, параграфів, таблиць, форм для введення даних, картинок, аудіо та відео елементів тощо. HTML документ складається із серії коротких тегів, записаних автором сайту в текстовий файл. Такий збережений текст зберігається як html файл та переглядається через певний веб-браузер, наприклад, Google Chrome чи Mozilla Firefox. Веб-браузер зчитує файл та переводить текст у видиму форму, відтворюючи дизайн сторінки. Визначення HTML складається із трьох наступних складових:

1. HyperText – спосіб, за допомогою якого користувач переходить від одної веб-сторінки до іншої, клікаючи на певні гіперпосилання. Той факт, що посилання мають префікс «гіпер» означає, що такий процес нелінійний: можна переходити від одної частини інтернету до іншої, перейшовши за будь-яким посиланням, без дотримання певного порядку переходу до іншої сторінки.
2. Markup пояснює що саме теги HTML роблять для показу інформації. Теги позначають структурні блоки сторінки, медіа, таблиці, певний тип тексту, наприклад, курсивний чи підкреслений тощо.
3. Language – HTML є мовою розмітки, тобто має власні визначені слова та синтаксис.

Не менш важливою технологією є CSS (Cascading Style Sheets) – інструмент, за допомогою якого розробник може визначати привабливі стилі відтворення елементів на веб-сторінці. Наприклад, стилі тексту, прозорість картинки, відступи навколо структурних блоків, розміри таблиць тощо. CSS надає веб-розробникам кращий контроль над тим, який вигляд будуть мати веб-сторінки, аніж розмітка HTML. CSS допомагає веб-розробникам створювати уніфікований дизайн декількох сторінок веб-сторінки: замість того, щоб визначати стиль кожної таблиці та кожного блоку тексту у HTML-розмітці сторінки, часто використовувані стилі

можуть бути визначені тільки один раз у документі css та багаторазово використані у майбутньому сторінками, що посилаються на певний css файл. Можливості CSS зростають із застосуванням таких компілюючих обробників шаблонів, як LESS чи Sass, що значно спрощує процес написання коду розробниками дизайну веб-сторінок.

JavaScript є потужним інструментом для розроблення front end частини веб-застосунку. JavaScript – це крос-платформна, об'єктно-орієнтована скриптова мова, що використовується для надання веб-сторінкам інтерактивності та динамічності, наприклад, створення складних анімацій, спливаючих меню тощо. JavaScript на стороні клієнта надає об'єкти та методи для керування браузером та його об'єктною моделлю документа (DOM). JavaScript код на стороні клієнта дозволяє динамічно розміщувати та змінювати HTML елементи, отримувати та обробляти дані форм, реагувати на такі дії користувача, як клацання миші, введення тексту чи навігація по сторінках. За допомогою JavaScript можна налагодити двосторонній зв'язок із серверною частиною веб-додатку, використовуючи технології Ajax та бібліотеку jQuery. За допомогою цих двох технологій можна динамічно оновлювати вміст веб-сторінки без фактичного оновлення сторінки користувачем, створювати запити до сервера, отримувати дані від сервера, надсилати дані до back end частини веб-додатку.

Бібліотеки JavaScript надають розробнику інтерфейс для візуалізації даних у вигляді інтерактивних графіків та різних видів діаграм. Прикладом таких бібліотек є D3.js, ChartJS, ThreeHS. D3.js є найпопулярнішою бібліотекою JavaScript для візуалізації даних за допомогою HTML, SVG та CSS. Ця бібліотека дозволяє візуалізувати дані у більш ніж 200 формах, що включають дерева, графіки на декартовій площині, картограми, гістограми, стовпчасті діаграми тощо.

Bootstrap – популярний фреймворк для створення front end частини веб-додатку. Цей фреймворк розроблений у Twitter та оснований на LESS CSS і jQuery. За допомогою Bootstrap можна реалізувати

адаптивний дизайн веб-сторінок, тобто такий дизайн сторінки, що буде змінюватись при зміні роздільної здатності екрану користувача. Bootstrap коректно працює із найпопулярнішими браузерами: Safari, Mozilla Firefox, Google Chrome, Opera, Microsoft Edge. Фреймворк надає зрозуміле та уніфіковане рішення для побудови веб-інтерфейсу розробниками. Цей фреймворк використовує сітку із 12 стовпців для розташування всіх елементів на сторінці веб-сайту, що спрощує процес побудови структури веб-сторінки. Також фреймворк цінний наперед визначеними стандартними стилями оформлення кнопок, форм, заголовків, що мають сучасний та привабливий вигляд, завдяки чому розробнику можна не приділяти багато зайвої уваги розробці власних CSS стилів.

2.4. Визначення тональності тексту

Аналіз тональності тексту, Sentiment Analysis – це автоматизований процес визначення ставлення автора щодо певного предмета у написаному тексті або розмові [30].

Через те, що кількість публічно доступної інформації в інтернеті постійно зростає, збільшується кількість текстів, що висловлюють певну думку, рецензій, записів на форумах, блогах, які містять корисну для бізнесу інформацію. Саме тому на даний момент аналіз настроїв у тексті є цікавою темою для дослідження. Такий аналіз має багато практичних застосувань: моніторинг ставлення публіки до бренду, оцінка рівня задоволеності обслуговування клієнтів, моніторинг ставлення соціальних медіа до певного предмета тощо.

Аналіз тональності є однією з областей напряму обробки природної мови (Natural Language Processing, NLP). Цей напрям описує системи, однією з можливостей яких є здатність ідентифікувати та вичленити оцінки тем, зазначених у тексті. Зазвичай такі системи аналізують текст на визначення таких параметрів:

1. Полярність – чи висловлює оратор позитивну або негативну думку.

2. Тема тексту – визначення того, про що говориться в тексті.
3. Автор думки – визначення особи або деякої сутності, що висловлює досліджувану думку.

Існує багато видів аналізу тональності тексту, а системи, що реалізують такий аналіз, поділяються на ті, що визначають полярність тексту (позитивну, негативну, нейтральну), що визначають почуття та емоції у тексті (сердиті, негативні, нейтральні) та що визначають наміри автора тексту (наприклад, наявність намір чи відсутність наміру купити продукт).

Аналіз полярності тексту дозволяє визначити точну оцінку ставлення автора до предмета в тексті. Замість того, щоб надавати оцінку «негативне» чи «позитивне», можна також розглядати такі більш точні категорії ставлень: дуже позитивне, позитивне, нейтральне, негативне, дуже негативне. Така оцінка може бути сприйнята як п’ятизірковий рейтинг: дуже позитивний текст відповідає п’ятьом зіркам, а дуже негативний – одній зірці. Також можна розглянути полярність тексту як неперервну величину, що знаходиться на відрізку [–1, 1], де –1 – найнегативніша полярність, +1 – найбільш позитивна полярність, а 0 – нейтральна полярність.

Існують декілька методів та алгоритмів для реалізації систем з аналізу тональності тексту, що можуть бути класифікованими таким чином:

1. Основані на визначених правилах – такі системи здійснюють аналіз на основі набору певних вручну визначених правил.
2. Автоматизовані – такі системи спираються на методи машинного навчання.
3. Гібридні – поєднують обидва підходи, описані вище.

Завдання автоматизованих методів зазвичай моделюється як задача класифікації, у якій класифікатору в якості вхідних даних надається нерозмічений текст, а як результат роботи класифікатора очікується передбачення відповідної категорії: позитивної, негативної чи нейтральної. Такий класифікатор, що створений на основі алгоритмів машинного

навчання, може бути реалізований у вигляді наступних компонентів та кроків, що показані на рис. 2.1.

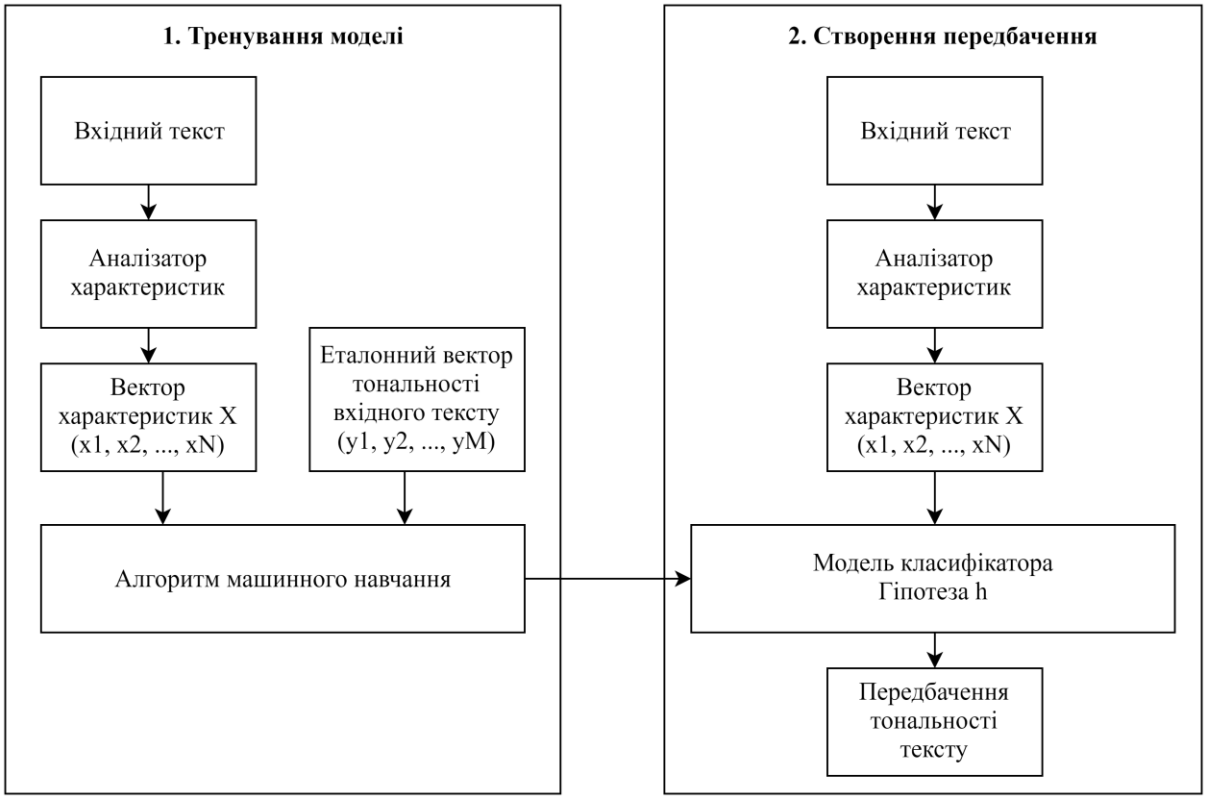


Рис. 2.1. Компоненти автоматизованої системи аналізу тональності тексту

Процес тренування моделі полягає у вивченні моделлю зв’язків між певним входом (текстом) та відповідної наперед визначеної еталонної тональності тексту на основі тестових зразків. Аналізатор характеристик перетворює вхідний текст у вектор характеристик. Пари визначених ознак та наперед визначених тональностей тексту подаються в алгоритм машинного навчання для створення моделі прогнозування та певної гіпотези, що найкраще описує існуючі та нові дані, що виникнуть в майбутньому.

У процесі прогнозування тональності тексту аналізатор характеристик використовується для перетворення текстових даних без наперед визначених тональностей у вектор характеристик. Варто зазначити, що на початку процесу передбачення не існує наперед визначеної

тональності, на відміну від початку процесу тренування. Наступним кроком вектори характеристик подаються у модель, яка генерує передбачені тональності: позитивні, негативні або нейтральні.

Крок класифікації зазвичай потребує реалізації однієї з таких статистичних моделей та алгоритмів:

1. Наївний баєсів класифікатор – сімейство ймовірнісних алгоритмів, які використовують теорему Баєса для прогнозування категорії тексту.
2. Логістична регресія – алгоритм у статистиці, що використовується для передбачення деякого значення при відомому наборі характеристик.
3. Методи опорних векторів – неймовірнісна модель, що використовує представлення прикладів текстів у вигляді точок у багатовимірному просторі. Ці приклади розташовані таким чином, що приклади різних категорій (настроїв) належать до різних областей простору. Нові тексти розташовуються на тому ж просторі, після чого проводиться передбачення категорії, до якої належить текст, у залежності від того, у яку область простору потрапляє новий текст.
4. Алгоритми глибинного навчання – різноманітний набір алгоритмів, що намагаються імітувати роботу мозку людини, використовуючи нейронні мережі для обробки даних.

У мові програмування Python існують рішення у вигляді бібліотек для коректної реалізації алгоритмів аналізу тональності тексту. Такі бібліотеки вирішують широкий спектр задач: визначення частин мови (Part-of-speech tagging), аналіз тональності, класифікація документів, моделювання теми тексту тощо. Одними із прикладів таких працюючих, добре задокументованих та готових до використання розробниками бібліотек є NLTK (Natural Language Toolkit), TextBlob, scikit-learn NLP toolkit та інші [31].

NLTK є найбільш відомою та повною NLP бібліотекою, має багато розширень, написаних спільнотою, має безліч готових підходів до багатьох задач NLP, реалізує швидку токенізацію речень, підтримує найбільшу

кількість мов у порівнянні з іншими бібліотеками. При застосуванні бібліотеки NLTK є можливість побудувати таку модель, що розуміє контекст та структуру речень, а не просто шукає певні слова. Завдяки такому підходу точність передбачення правильної тональності текстів збільшиться.

Для оцінки тональності тексту можна використати модель «Мішок слів» (Bag-of-words model). Ця модель повністю фокусується на словах чи на словосполученнях, але не звертає уваги на контекст речення та не намагається працювати із фундаментальними особливостями мови. Модель «Мішок слів» створює великий словник, який містить у собі пари «слово-тональність», а також кількість повторень кожного слова в реченні. Результатом оцінки тональності всього речення є зважена сума тональностей усіх вибраних слів у реченні. Бібліотека scikit-learn NLP toolkit містить функції для коректної реалізації моделі bag-of-words для вирішення задач класифікації текстів, надає можливість побудови власних моделей машинного навчання, має докладну документацію та інтуїтивні методи класів.

Метод «Мішка слів» має свої обмеження:

1. Семантичне значення слів (значення слова, контекст у реченні, порядок появи у реченні) не враховується.
2. Розмір вектора може бути величезним у випадку великих текстів, в результаті чого збільшиться об’єм оперативної пам’яті програми, а також збільшиться час виконання обчислень над вектором.

Бібліотека TextBlob побудована на основі бібліотек NLTK та Pattern. Ця бібліотека спрощує процес обробки тексту, надаючи зрозумілий інтерфейс для користування, є чудовим вибором для створення прототипу NLP моделі, має інструменти для токенізації речення, виділення N-грам, приведення слів до початкової форми однини називного відмінка, роботи із декількома мовами, аналізу тональності тексту як на основі вбудованих інструментів, так і на основі власноруч визначеного класифікатора.

Ще однією бібліотекою, що створена для аналізу тональності текстів, є VADER (Valence Aware Dictionary and sEntiment Reasoner), пакет vaderSentiment [32]. VADER – інструмент для аналізу тональності тексту, що спеціально налаштований на роботу із текстами повідомлень соціальних медіа, оскільки був натренований на повідомленнях Twitter, відгуках про товар на сайті amazon, відгуках про фільми та записах газети New York Times [33]. Такий набір для тренування моделі вказує на те, що VADER має справлятись із короткими текстовими повідомленнями, що мають «людський» вигляд: можуть містити друкарські помилки, сарказм, скорочення, відсутність дотримання правил правопису. VADER може працювати навіть із великими текстами, власноруч розділяючи їх на частини, а також ця бібліотека працює значно швидше, ніж її аналог TextBlob.

Бібліотека VADER створює складену оцінку (compound score) досліджуваного тексту – показник, що є сумою полярностей усіх слів, які були нормалізовані до значень від –1 (найбільш негативна полярність) до +1 (найбільш позитивна полярність). Ця нормалізована метрика є достатньо точною для визначення позитивності чи негативності ставлення автора до предмета у тексті. На таку оцінку тональності також впливають регістр букв, кількість знаків оклику, смайли. Наприклад, речення (у перекладі на українську мову) «Нова компанія має потенціал» та «Нічого собі!!Ці хлопці ОБВАЛЯТЬ нішу!:)» мають різну оцінку тональності, оскільки у другому реченні використовувалось повторення знаку «!», а також був наведений позитивний смайл.

3. ОГЛЯД РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

3.1. Загальна структура системи

Програмне забезпечення реалізоване у вигляді веб-додатку. Таку розроблену програмну систему можна умовно поділити на дві умовні частини: клієнтська частина (front end) та серверна частина (back end). Кожна із цих частин виконує свою роль, має своїх користувачів та виконує свої функціональні та нефункціональні вимоги.

Серверна частина розроблюваного програмного забезпечення виконує такі функції:

1. Оброблення асинхронних запитів від front end частини веб-додатку.
2. Реалізація архітектурного патерну MVC.
3. Реалізація роботи із базою даних для зчитування, збереження, оновлення та видалення даних.
4. Щогодинний збір фінансових даних криптовалют, їх фільтрація та збереження зібраної інформації у базу даних.
5. Збір даних новин соціальних новин та преси, їх оброблення та збереження текстів у базу даних.
6. Проведення аналізу тональності текстів за допомогою алгоритмів машинного навчання.
7. Передбачення курсу вибраних криптовалют.

Клієнтська частина у свою чергу виконує наступні функції:

1. Реалізує інтерфейс для взаємодії системи із користувачем
2. Формує асинхронні запити до сервера для динамічного оновлення даних на веб-сторінці.
3. Реалізує візуалізацію отриманих передбачень у вигляді графіків та діаграм.
4. Відповідає за дотримання коректності відтворення дизайну сторінки на пристроях із різною роздільною здатністю екрану.

Структурна схема системи (див. Додаток 1) розробленого додатка складається із багатьох елементів, кожен з яких є пов’язаний із іншим модулем у системі та має свою наперед визначену роль. Існуючі елементи на схемі поділяються на такі типи: модулі системи, база даних, користувачі системи.

У розробленому програмному забезпеченні існують такі модулі:

1. Модуль збору текстових даних. Даний модуль збирає повідомлення від соціальної мережі Twitter, форуму Reddit та сайтів новин та передає необроблені дані іншому модулю, що очистить зібрані дані.
2. Модуль аналізу повідомлень медіа. Даний модуль проводить аналіз тональності зібраних необроблених текстів, попередньо очистивши та привівши до очищеної форми зібрані тексти. Результати проведеного аналізу передаються у основний модуль, що має доступ до бази даних, для їх подальшого збереження.
3. Модуль збору фінансових даних. Даний модуль збирає дані про ціни досліджуваних цифрових валют, а також про фінансові індекси загального криптовалютного ринку. Зібрані дані передаються у основний модуль для подальшого збереження у базі даних.
4. Модуль передбачення фінансових даних. Даний модуль створює передбачення курсу певної криптовалюти на досліджувані інтервали. Створене передбачення передається у основний модуль, що зберігає їх у базі даних.
5. Модуль API для сторонніх розробників. Цей модуль забезпечує доступ до певних результатів роботи системи стороннім розробникам.
6. Модуль зв’язку із базою даних. Даний модуль використовує драйвер, за допомогою якого сервер має можливість зчитувати, зберігати, оновлювати та видаляти записи у базі даних.
7. Основна серверна частина. Модуль серверної частини є центральним, а тому основним, оскільки цей модуль пов’язує усі інші модулі із базою даних, а також опрацьовує усі види даних від інших модулів.

8. Клієнтська частина. Даний модуль відповідає за подання результатів роботи системи користувачеві на веб-сторінках.

Варто зазначити, що модуль «Server View», що відповідає за відтворення результатів роботи системи користувачу, використовує модуль «Server API», що допомагає оновлювати дані у веб-браузері користувача без перезавантаження усієї сторінки.

UML діаграма компонентів (див. Додаток 1) показує існуючі компоненти у системі, а також внутрішні і зовнішні інтерфейси, по яким компоненти спілкуються між собою. На діаграмі виділені дві окремі частини: back end та front end частини додатку. Front end частина зв’язана із серверною частиною через інтерфейс RESTful API. Видно які саме інтерфейси необхідні компонентам: компонент збору фінансових даних потребує API сервісів, що надають такі фінансові дані, компонент аналізу текстів потребує інтерфейс, що забезпечить текстовими даними, а компонент, що надає доступ до бази даних, потребує інтерфейс драйвера psycopg для підключення до бази даних PostgreSQL. На діаграмі також видні залежності між інтерфейсами всередині серверної частини веб-додатку: інтерфейс, що створює прогноз ціни, залежить від результатів роботи інтерфейсу, що забезпечують фінансові показники цифрових валют, а також інтерфейсу, що надає тональність тексту.

Розроблений програмний застосунок містить API, що надає стороннім розробникам можливість використовувати результати роботи системи у власних проектах. Використовуючи цей API, програмісти мають можливість:

1. Отримувати список останніх новин про певну криптовалюту.
2. Отримувати передбачення досліджуваних криптовалют та усього криптовалютного ринку.
3. Отримувати інформацію про тренди досліджуваних криптовалют та тренд узагальненої ринкової капіталізації.

4. Отримувати загальну інформацію про досліджувані цифрові валюти та ресурси новин та соціальні мережі.
5. Отримувати історичні фінансові дані та історичні дані тональностей тексту.

Основною мовою програмування для розроблення back end частини веб-додатку було вибрано Python, оскільки у проекті має бути велика кількість обчислень на процесорі, а також тому, що у програміста розроблюваного програмного забезпечення є достатній досвід саме у Django, веб-фреймворку, написаному мовою програмування Python.

Для веб-додатка розроблена схема реляційної бази даних, структура якої зображена у вигляді entity-relation діаграми (див. Додаток 1). Дана діаграма описує таблиці бази даних та типи зв'язків між ними, зовнішні та первинні ключі, типи полів таблиць. Усі типи полів, зазначені на діаграмі, є визначеними у реляційній СКБД PostgreSQL.

У створеній базі даних визначені такі таблиці: Market, Currency, Media, Price, Prediction, NewsMedia, TwitterMedia, RedditMedia, User, UserRole, Role. Усі таблиці нормалізовані у першу, другу та третю нормальну форму, тобто створено окремі таблиці для кожного набору даних, де дані визначені первинними ключами, було усунено однакові дані в таблицях, створені окремі таблиці для наборів даних, що відносяться до декількох записів, а ці таблиці у свою чергу пов'язані один з одним за допомогою зовнішніх ключів, було видалено поля, що не залежать від жодного ключа [34]. Для таблиць Price та Prediction було створено індекси для полів, що позначають дату певної ціни або передбачення, для пришвидшення виконання запитів, що використовують поле дати.

Таблиця Media зберігає загальну інформацію про джерела досліджуваних текстів новин:

1. Пряме посилання на домашню сторінку інформаційного ресурсу.
2. Назву ресурсу.
3. Скорочений опис ресурсу.

4. Розширений опис ресурсу.

Таблиця Market зберігає узагальнену погодинну характеристику криптовалютного ринку:

1. Дата та час певної точки на часовому проміжку.
2. Усереднена ринкова капіталізація досліджуваних криптовалют за останню годину, в USD.
3. Усереднений обсяг торгівлі криптовалют на більшості бірж, в USD.
4. Нормалізована оцінка тональності повідомлень відстежуваних новин про загальний ринок за попередню годину.
5. Передбачення ринкової капіталізації на наступну годину, в USD.
6. Нормалізована оцінка тренду ринкової капіталізації: підвищення або зниження загальної капіталізації на наступну годину.

Таблиця Currency зберігає загальну інформацію про досліджувані цифрові валюти, що торгуються на біржах:

1. Символ криптовалюти для торгівлі на криптобіржі.
2. Назва криптовалюти.
3. Короткий опис криптовалюти.
4. Розширений опис цифрової валюти.

Таблиця Prediction зберігає погодинні значення передбачень котирувань певної криптовалюти, в USD:

1. Зовнішній ключ, що вказує на належність передбачення до певної криптовалюти.
2. Дата та час певної точки на розглянутому часовому проміжку.
3. Передбачення ціни на наступну годину.
4. Передбачення ціни через 24 години.
5. Передбачення ціни через 1 тиждень.
6. Оцінка тональності новин, пов’язаних із розглянутою цифровою валютою, за останню годину, добу, місяць.
7. Нормалізована оцінка тренду курсу криптовалюти.

Таблиця Price зберігає погодинні значення ціни певної криптовалюти в узагальненому криптовалютному ринку, в USD:

1. Зовнішній ключ, що є ідентифікатором розглянутої криптовалюти.
2. Ціна валюти на початку години.
3. Ціна валюти на кінці години.
4. Найвища ціна протягом години.
5. Найнижча ціна протягом години.
6. Різниця між найвищою та найнижчою ціною на дану годину.
7. Різниця між ціною на початку години та під кінець розглянутої години.

Таблиця NewsMedia зберігає інформаційні повідомлення від досліджуваних сайтів новин. Ця таблиця посилається на таблиці Media та Currency обов'язковим зв'язком один-до-одного для ідентифікації того, про яку криптовалюту йдеться мова на якому сайті новин. Завдяки цій таблиці можна дізнатись такі відомості про зібрані новини:

1. Пряме посилання на статтю чи повідомлення, що досліджується.
2. Заголовок статті.
3. Повний, необроблений для алгоритму аналізу тональності, текст статті.
4. Дату публікації статті.
5. Оцінку тональності тексту алгоритмом бібліотеки TextBlob.
6. Оцінку тональності тексту алгоритмом бібліотеки VADER.
7. Оцінку тональності тексту власним алгоритмом, що показав найкращу точність передбачень.

Таблиця TwitterMedia зберігає текстові записи користувачів соціальної мережі Twitter. У цій таблиці зберігаються такі дані про твіти:

1. Ідентифікаційний номер повідомлення у системі Twitter.
2. Текст повідомлення, включаючи смайли та посилання.
3. Кількість лайків, ретвітів (публікація повідомлення на власній сторінці) та відповідей на повідомлення.

4. Дата публікації повідомлення.
5. Оцінка тональності тексту алгоритмами із бібліотек TextBlob, VADER та власним алгоритмом, що дав найкращу оцінку передбачення.

Таблиця RedditMedia зберігає інформацію про досліджувані розділи інтернет-форуму Reddit, так звані сабреддіти. Зберігається така інформація:

1. Заголовок повідомлення.
2. Оцінка користувачами повідомлення – кількість «upvotes».
3. Кількість коментарів.
4. Пряме посилання на повідомлення.
5. Дата публікації.
6. Оцінка тональності заголовка від трьох досліджуваних алгоритмів.

У розробленій програмній системі існують таблиці, що описують профілі зареєстрованих користувачів: User, UserRole, Role. Ці таблиці містять таку інформацію про облікові записи:

1. Ім’я та прізвище користувача.
2. Логін у системі.
3. Пароль профілю.
4. Позначка того, чи є користувач адміністратором.
5. Позначка того, чи є користувач у системі онлайн.
6. Дата останнього входу у систему.
7. Дата створення облікового запису.

3.2. Алгоритм оброблення вхідних даних

У розробленій системі існує декілька типів вхідних даних:

1. Текстові дані: дані повідомлень соціальної мережі Twitter, форуму Reddit, веб-сайтів новин виключно криптовалютної тематики на кшталт Coindesk, CCN, CoinTelegraph, сайтів новин бізнес-тематики на кшталт Forbes, CNBC.

2. Фінансові дані: погодинні дані цін таких криптовалют від відкритого ресурсу Cryptocompare: Bitcoin (BTC-USD), Ether (ETH-USDT), Litecoin (LTC-USD).

Вхідні дані можна також поділити на 2 типи за часом їх збереження та оновлення у системі:

1. Архівні дані: тексти постачальників новин та дані цін розглянутих криптовалют за минулий проміжок часу, що починається від 2016-01-01 та закінчується днем останнього запуску програми. Такі дані оновлюються вручну один раз адміністратором системи. Архівні дані необхідні для роботи системи, оскільки для коректної роботи алгоритму передбачення необхідно мати достатньо великий набір даних.
2. Свіжі дані: тексти та фінансові показники, що оновлюються кожну годину та зберігаються як архівні по приході нових свіжих даних. Такі дані оновлюються щогодинно, автоматично та автономно. Дані одразу зберігаються у базу даних, із приходом нових даних оновлюється стан системи.

Для отримання фінансових показників криптовалют необхідно отримати та розшифрувати відповідь від певної функціональної точки RESTful API Cryptocompare. Спершу створюється запит, у якому вказується необхідний часовий проміжок у форматі UNIX timestamp – кількість секунд від 1970-01-01 00:00:00 UTC. Відповідь на цей запит приходить на сервер у форматі JSON, що може бути представлений у вигляді структури даних словник. Необхідні дані із отриманого словника зберігаються у таблиці Price, непотрібні дані видаляються із оперативної пам’яті. API Cryptocompare має свої обмеження: не більше 100,000 запитів на місяць від однієї IP-адреси.

Для отримання фінансових показників узагальненого ринку криптовалют використовується неофіційний ресурс coinmarketcap.com. Під неофіційним ресурсом мається на увазі той ресурс, що не представлений у

відкритому доступі та неналежне використання якого може призвести до перманентного блокування IP-адреси. Для отримання даних показників ринкової капіталізації та обсягу торгівлі необхідно надіслати запит за певною адресою, вказавши необхідний часовий проміжок. Відповідь на цей запит приходить на сервер у форматі JSON, у якому система знаходить та зберігає необхідні дані у таблиці Market.

Оскільки збір фінансових даних проходить із допомогою сторонніх сервісу, розроблюване програмне забезпечення має слідкувати за повнотою збору даних та цілісності інформації, тобто чи відбувався успішний збір та збереження даних кожну годину роботи програми. Для цього кожні 24 години проходить перевірка, у ході якої відбувається повторний збір відсутньої або пошкодженої інформації за розглянутий проміжок часу. Дану перевірку може запустити адміністратор системи, використовуючи панель адміністратора.

Текстові дані повідомлень соціальних мереж та статей сайтів інтернет-новин мають різний формат, тому потребують різні підходи для своєї обробки та подальшого збереження. У зборі текстових даних новин є спільне: у базу даних зберігаються нерозмічені тексти у початковому вигляді, оскільки є необхідність повторного аналізу текстових даних, наприклад, новим алгоритмом аналізу тональності тексту. Ще одним спільним фактом є перевірка цілісності та повноти текстових даних кожні 24 години та виправлення у разі знаходження помилок.

Необроблені дані у базі даних мають бути певним чином обробленими та відформатованими для того, щоб алгоритм визначення тональності тексту працював якнайкраще. Тому модуль очищення текстових даних уніфікує усі типи досліджуваних текстів: короткі замітки із Twitter, єдиний вигляд – вектор. Процес очищення текстових даних виконується кожну годину, оскільки відформатовані дані необхідні для щогодинного оновлення передбачення курсу криптовалюти.

Повний універсальний алгоритм очищення зібраних текстових даних наведений на плакаті «Алгоритм аналізу повідомлень соціальних медіа» (див. Додаток 1). На вхід алгоритму подаються необроблені текстові дані. На вихід алгоритм надає очищені текстові дані, що не містять великого рівня шуму для алгоритмів визначення тональності тексту. Процес очищення містить такі кроки:

1. Видалення продубльованих та пустих записів.
2. Видалення нерелевантних записів, наприклад, автоматично згенерованого контенту або статей, що не стосуються ринку.
3. Видалення HTML-розмітки.
4. Декодування тексту у формат UTF-8.
5. Заміна слів із апострофом: «you’re» перетворюється на «you are».
6. Видалення стоп-слів.
7. Видалення URL-адрес.
8. Видалення запитів, що написані не англійською мовою.
9. Приведення його до форми однини називного відмінка.
10. Видалення занадто коротких записів (менше ніж 5 символів).
11. Токенізація речення – перетворення тексту у вектор слів.

Після завершення процесу очищення текстових даних необхідно знайти та зберегти словник специфічних для сфери криптовалютного трейдингу сленгу. Наприклад, слово «tothemoon» означає «підвищення ціни криптовалюти». Наступним кроком усі залишені повідомлення були збережені та проаналізовані на частоту виявлення хештегів, слів, найчастіше вживаних *N*-грам.

Експериментальними варіантами очищення текстових даних були наступні операції:

1. Видалення зайвих голосних букв, що повторюються якнайменше 3 рази підряд, наприклад, слово «cooooool» має бути перетворене на слово «cool».

2. Заміна тексту, що імітує сміх. При обробці даних було помічено, що багато повідомлень соціальних мереж Twitter та Reddit містять слова, що імітують сміх: «хахахаха» та «ahahahaha» – такі слова замінені на слово «laugh».
3. Конвертація емодзі у текстовий вигляд, що точно показують тональність цих символів, наприклад, емодзі «:)» перетворений у словосполучення «smiling happy».

3.3. Алгоритм аналізу повідомлень соціальних медіа.

Розроблена програма застосовує декілька алгоритмів для оцінки тональності тексту для подальшого порівняння та вибору найкращого підходу серед поданих. Після реалізації алгоритмів перевіряється точність їх передбачення та інші метрики. Серед застосованих та протестованих алгоритмів є такі:

1. Бібліотека TextBlob.
2. Бібліотека VADER.
3. Наївний баєсів класифікатор та метод «Мішок слів».

Застосовані алгоритми використовують на вході очищені текстові дані, а на вихід створюють оцінку тональності тексту (див. Додаток 1).

Наведені вище методи були застосовані до усіх зібраних записів із Twitter, Reddit та сайтів новин. Для цього модуль аналізу текстових даних використовує інтерфейс модулю роботи із основною базою даних для отримання та оновлення даних.

Бібліотеки TextBlob та VADER мають можливість розширювати лексикон досліджуваних слів. Розглянуті у системі текстові дані соціальних медіа містять багато сленгових слів, скорочень та таких слів, що використовуються лише у сфері криптовалютного трейдингу («tothemoon»). Для покращення роботи цих двох бібліотек на основі 10,000 повідомлень соціальної мережі Twitter було створено та використано додатковий словник спеціальної лексики, що вміщує 540 слів. У результаті цього

точність передбачень зросла на +2.41% та +1.61% для алгоритмів бібліотек TextBlob та VADER відповідно.

Після отримання усіх оцінок текстів можна створити хмару тегів, що відображає найбільш вживані слова, а також дослідити найбільш уживані словосполучення двох та трьох слів: біграми та триграми, що допоможе користувачам системи краще зрозуміти вплив розглянутих у майбутньому повідомлень.

3.4. Алгоритм прогнозування курсу криптовалют

Алгоритм прогнозування фінансових даних досліджуваних криптовалют потребує на вхід такі дані: тренд курсу криптовалюти, сумарна тональність повідомлень за певний інтервал часу, а на вихід надає прогноз на певний інтервал часу: на наступну годину, через 24 години та через тиждень, тобто через 168 годин (див. Додаток 1).

Усі вхідні дані комбінуються у єдиний набір даних у вигляді єдиного pandas Dataframe. До цих даних додаються наступні, додаткові характеристики (features), що покращить роботу алгоритмів передбачення:

1. Відношення сумарної тональності текстів на попередньому інтервалі до поза попереднього інтервалу.
2. Середній об’єм торгівлі криптовалюти за минулий ковзний інтервал.
3. Середня ціна криптовалюти за минулий ковзний інтервал.

Отримані дані діляться по інтервалах: погодинні, щодобові, щотижневі, та по типах криптовалют: Bitcoin, Ether, Litecoin. Поділені дані також зберігаються у вигляді Dataframe.

Наступним кроком усі поділені дані розбиваються на набір для тренування та набір для тестування. Набір для тренування використовується для тренування моделі передбачення та отримання гіпотези, а набір для тестування для валідації результатів роботи отриманої гіпотези.

Процес тренування проходить у наступних алгоритмах:

1. Логістична регресія.

2. Поліноміальний баєсів класифікатор.
3. Ансамблевий метод Random Forest.

Після проведення валідації було проведено аналіз кореляції між характеристиками набору даних, побудовано матрицю неточностей, а також визначено оцінки точності алгоритмів машинного навчання.

Модель та її результати було збережено за допомогою модулю pickle у бінарному форматі для подальшого аналізу та можливого покращення моделі за рахунок більш точних параметрів, або оновлених даних.

Отримані моделі було застосовано до усіх фінансових даних, а також збережено у базі даних у таблиці Prediction.

3.5. Модуль формування звітів

Розроблювана програмна система передбачає створення модулю для формування звітів із проміжними та фінальними результатами роботи системи.

Модуль формування звітів та візуалізації роботи веб-додатку використовує модуль бази даних

Такі звіти у створеному веб-додатку розміщуються у вигляді графіків та діаграм, що візуалізують такі дані:

1. Історію ціни криптовалют Bitcon, Litecoin, Ether від 2016-01-01 по день запуску програми.
2. Передбачення ціни досліджуваних криптовалют на наступну годину, добу та місяць.
3. Історію та поточні показники тональності повідомлень соціальних медіа та сайтів новин щодо криптовалют та узагальненого криптовалютного ринку.
4. Історію узагальненої капіталізації ринку криптовалют від 2016-01-01 по день запуску програми.
5. Передбачення капіталізації ринку на наступну годину.

Веб-додаток також надає можливість переглянути список останніх досліджених твітів, записів Reddit та найвпливовіших записів новин у формі списку. Ця можливість доступна лише довіреним особам, що зареєстровані адміністратором системи. У цьому списку видно не тільки досліджений текст, а й оцінку тональності тексту реалізованими алгоритмами.

4. АНАЛІЗ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості реалізації

Back end частина веб-додатку реалізує архітектурний дизайн MTV (Model Template View) веб-фреймворку Django. MTV – одна із різновидностей патерну MVC (Model View Controller). У даній архітектурі частина Model (models.py) містить логічну структуру даних та є посередником між базою даних та частиною View. Частина View (views.py) форматує дані за допомогою частини Model, спілкується із базою даних та оброблює дані, що передаються у частину Template. Частина Template оброблює усе, що показує веб-браузер [35].

Процес роботи MTV архітектури розробленого додатка показаний на рис. 4.1. На цьому рисунку показана взаємодія частин Model, View та Template. Зображено, що частини Model та View знаходяться у серверній частині, а частина Template – у клієнтській частині. Template може спілкуватись із серверною частиною через частину View, що, у свою чергу, виконує певні операції та отримує основні дані від бази даних від Model.

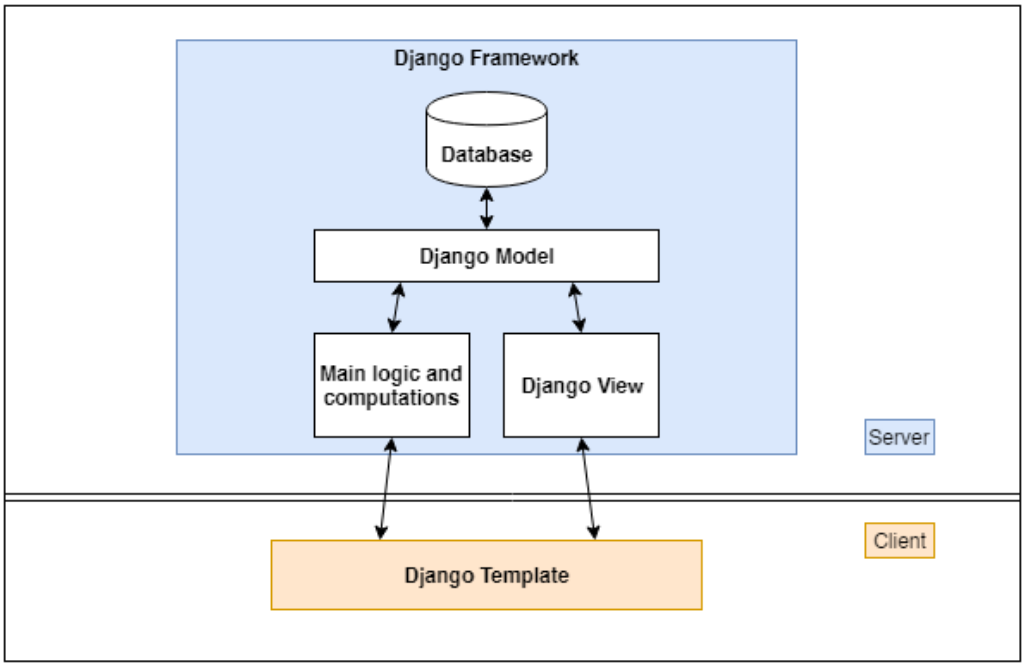


Рис. 4.1. Процес роботи архітектури MTV веб-фреймворку Django

Для масштабованості системи реалізовано декілька шаблонів проектування програмного забезпечення: «Будівельник», «Адаптер» та «Декоратор». Патерн «Будівельник» використаний як сутність, що допомагає уніфікувати процес покроковий процес аналізу тексту: від збору та очищення даних до серіалізації проаналізованих даних у JSON форматі. Даний патерн допомагає використовувати один і той же програмний код для аналізу різних за своєю суттю ресурсів, наприклад, сайту новин та соціальної мережі Twitter. Шаблон проектування «Адаптер» використаний для проведення аналізу тональності текстів різних форматів: необроблений текст, вектор оброблених слів. Адаптер «Декоратор» у програмному забезпеченні допомагає додавати функціональність методам, що оброблюють запити до функціональних точок API. Прикладом такої функціональності є обмеження максимальної кількості запитів за хвилину.

4.2. Дизайн та вміст веб-сторінок

Для реалізації дизайну та вмісту сторінок веб-додатку було використано технології HTML5, CSS та JavaScript. Усі сторінки веб-додатку мають дизайн, що підлаштовується під роздільну здатність екрана користувача – адаптивний design. Такий дизайн реалізовано завдяки бібліотеці Bootstrap 4. При створенні сторінок було використано технологію Django template language – мову розмітки, що дозволяє визначити шаблони для веб-сторінок. Таким чином, було створено базовий шаблон сторінки, що містить універсальні для усіх сторінок HTML елементи (header, footer, включення основних CSS та JavaScript файлів, підключення CDN бібліотек). Такий базовий шаблон був використаний для реалізації усіх інших сторінок веб-додатку.

У веб-додатку існують такі сторінки:

1. Головна сторінка.
2. Сторінка про криптовалюти.
3. Сторінка про криптовалютний ринок.

4. Сторінка останніх зібраних новинних записів.
5. Сторінка оновлення системних даних.
6. Сторінка входу до системи.
7. Сторінка реєстрації довіреної особи.
8. Сторінка керівництва користувача.
9. Сторінка керівництва програміста.
10. Сторінка про призначення системи.
11. Сторінка про роботу алгоритмів.

Одними із універсальних елементів, що присутні на кожній сторінці веб-додатку, є блок навігаційної частини (див. рис. 4.2) та блок у нижній частині сторінки, footer (див. рис. 4.3). Використовуючи ці елементи, користувач веб-додатку має можливість пересуватись по гіперпосиланнях, що ведуть на сторінки, що розповідають про систему або візуалізують результати роботи системи. Наприклад, переглянути передбачення криптовалют та узагальненого ринку цифрових можна перейшовши за посиланням на навігаційній частині веб-сторінки, а переглянути керівництво користувача або почитати про призначення системи можна перейшовши за посиланнями, зазначеними у нижній частині веб-сторінки.

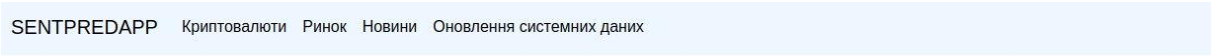


Рис. 4.2. Універсальна навігаційна частина веб-сторінки

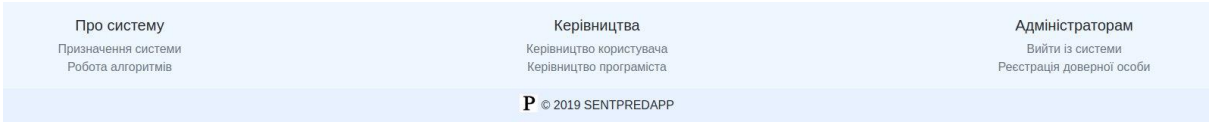


Рис. 4.3. Універсальна нижня частина веб-сторінки

Головна сторінка містить короткий опис системи та посилання на сторінки, що відтворюють основні результати роботи системи. На головній

сторінці можна також перейти до сторінки входу до системи адміністратором або довіреною особою.

Сторінка про криптовалюти зображена містить інформацію про три досліджувані цифрові валюти: Bitcoin, Litecoin, Ether. Для кожної цифрової валюти наведено окремий графік, що містить інформацію про ціну валюти, передбачення її ціни та середню тональність текстів. Над графіком зображено корисну фінансову інформацію: зміну ціни за 24 години, об’єм продажу, сумарна капіталізація тощо.

На рис. 4.4 зображено знімок екрану сторінки, що описує узагальнений криптовалютний ринок.

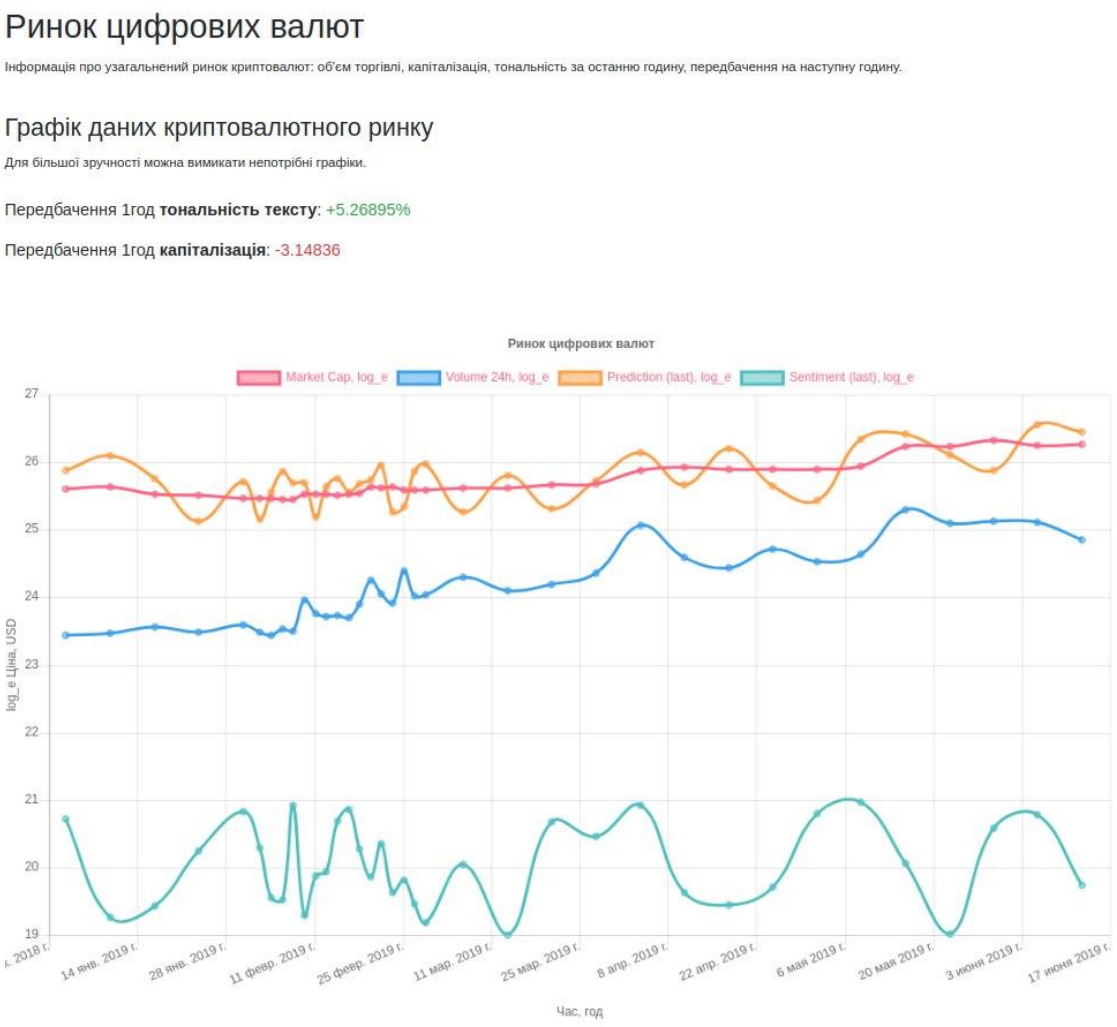


Рис. 4.4. Сторінка опису криптовалютного ринку

На сторінці опису криптовалютного ринку показано розраховане передбачення капіталізації на наступну годину та розрахований прогноз тональності тексту на наступну годину. На графіку даних зображена інформація від 2018-01-01 по сьогоднішній день про загальну капіталізацію, об’єм торгівлі за 24 години, передбачення капіталізації, передбачення тональності текстів про ринок цифрових валют. Графік є інтерактивним: можна видаляти непотрібні графіки та передивлятись значення у деякій точці на графіку, навівши курсором миші на необхідну точку.

Веб-додаток містить сторінки, на які можливо потрапити лише аутентифікованим користувачам. Такими сторінками є сторінка створення довіреної особи, сторінка перегляду останніх зібраних новин та сторінка оновлення системних даних. При переході за адресою до будь-якої із цих сторінок неавтентифікованим користувачем, йому буде запропоновано увійти до системи.

Сторінка входу до системи зображена на рис. 4.5. На цій сторінці користувачу пропонується ввести пару значень логін та пароль для входу до системи.

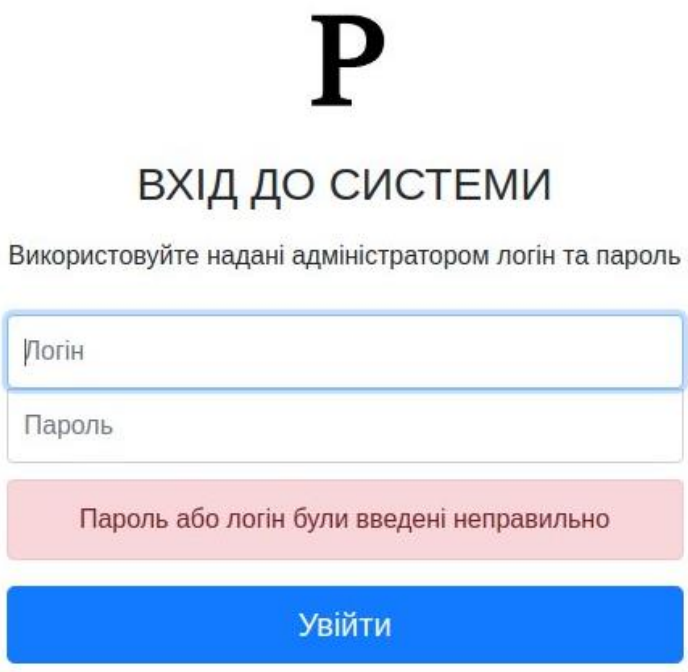


Рис. 4.5. Сторінка входу до системи

При введенні невірних даних буде виведено повідомлення про помилку, що показано на рис. 4.5. При введені коректних даних користувача буде перенаправлено до головної сторінки системи.

Користувач, що увійшов у систему як «довірена особа», має право переглядати останні зібрані новини на сторінці, зображеної на рис. 4.6. На такій сторінці зображені останні 20 повідомлень від кожного із новинних ресурсів: Twitter, Reddit та інші вибрані сайти новин. Кожне повідомлення від соціальних медіа містить пряме посилання на саме повідомлення, текст, що необхідний для дослідження на тональність, дату публікації повідомлення та оцінку тональності від трьох досліджуваних методів визначення тональності тексту: TextBlob, VADER та власний класифікатор.

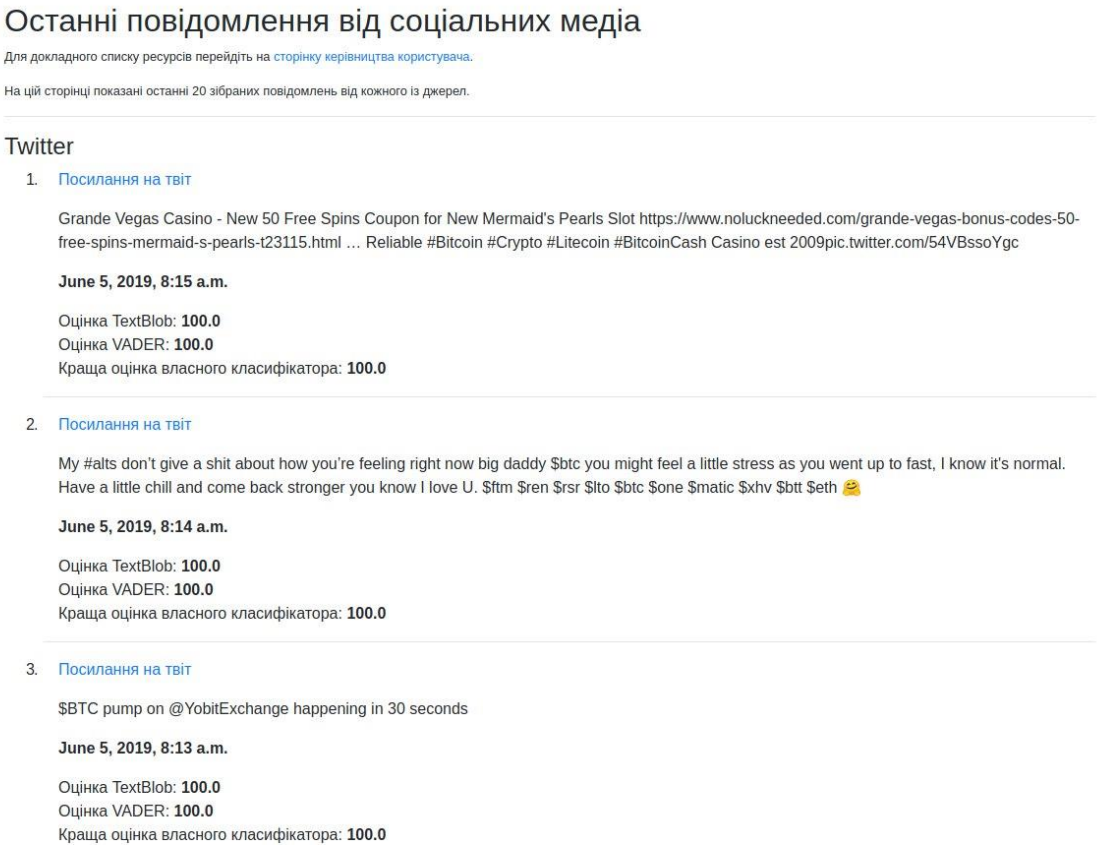


Рис. 4.6. Сторінка показу останніх повідомлень від соціальних медіа

Адміністратор веб-додатку має право створювати довірених користувачів на сторінці «Реєстрація довіреної особи», знімок екрану якої

показаний на рис. 4.7. Доступ до цієї сторінки має лише адміністратор ресурсу. На цій сторінці адміністратору ресурсу пропонується ввести поля, що необхідні для створення нової довіреної особи: логін і пароль для входу, а також підтвердження паролю.

При успішному створенні нового користувача під формою виводиться повідомлення результату із іменем нового користувача, що показано на рис. 4.7. При неуспішному створенні на місці повідомлення про успіх виводиться повідомлення про помилку. Поле логін користувача має обмеження на свій вміст: максимум 150 символів, можна використовувати лише латинські букви, цифри та деякі символи. Поле паролю має містити понад 8 символів, а також не має містити лише загальновживані слова.

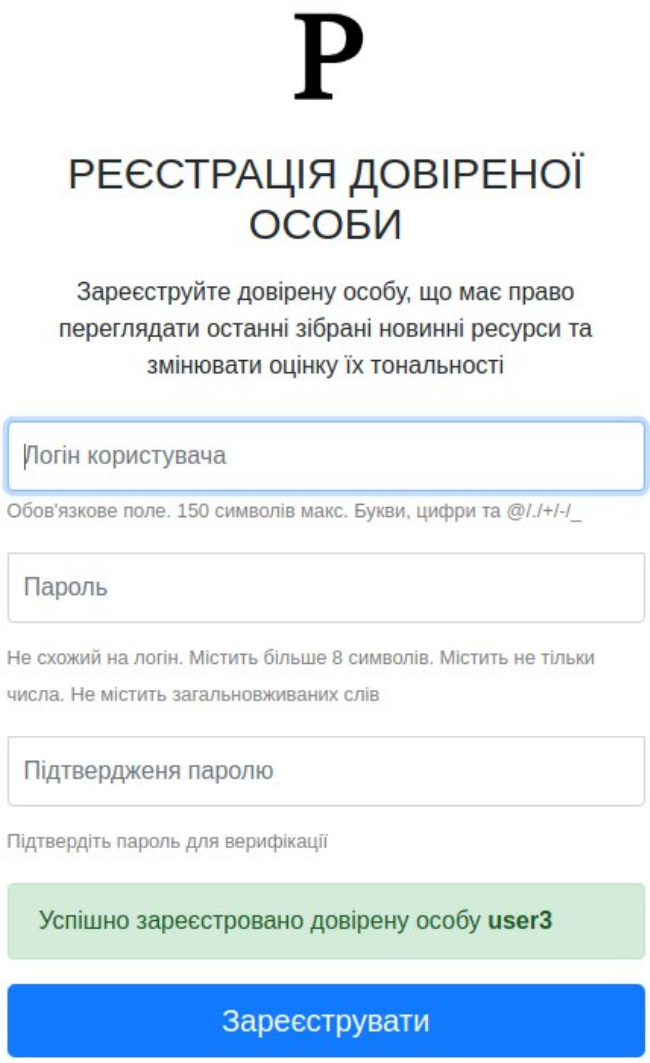


Рис. 4.7. Сторінка реєстрації довіреної особи

Адміністратор ресурсу має можливість вручну змінювати системні дані: оновлювати історичні фінансові та текстові дані на сторінці «Оновлення системних даних», що показана на рис. 4.8. На цій сторінці можна оновити як за останній час (приблизно 1 тиждень), так і за будь-який часовий проміжок від 2016-01-01. При натисненні будь-якої кнопки на сторінці з'являється колесо, що крутиться, тим самим показуючи той факт, що система виконує оновлення даних. При успішному зборі даних на місці колеса з'являється повідомлення про успішно завантажені дані, а при неуспішному зборі даних – повідомлення про помилку.

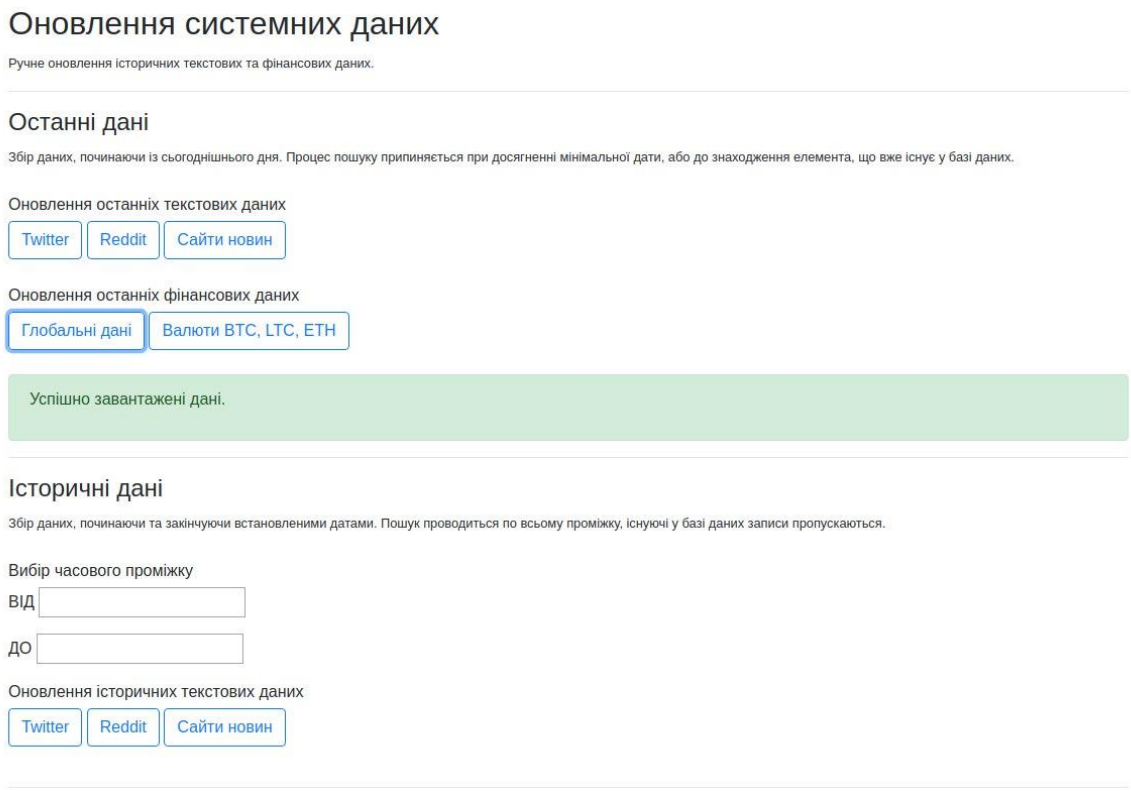


Рис. 4.8. Сторінка оновлення системних даних

На сторінці керівництва користувача указана корисна інформація та поради для найефективнішого користування веб-додатком. На сторінці про призначення системи наведено інформацію про основну функціональність системи, а також зазначено розробника програмного забезпечення. На сторінці про роботу системи наведено інформацію про основні алгоритми

для визначення тональності тексту та створення передбачень значень часового ряду, що використовуються у веб-додатку.

Веб-система надає вільний доступ до свого API стороннім розробникам. Правила використання такого програмного інтерфейсу наведені на сторінці опису API системи, знімок екрану якої показаний на рис. 4.9. Також наведено посилання на вихідний код системи та загальну інформацію про обмеження частоти запитів до функціональних точок системи. Для кожного запиту наведено його опис та приклад відповіді серверної частини програмного забезпечення.

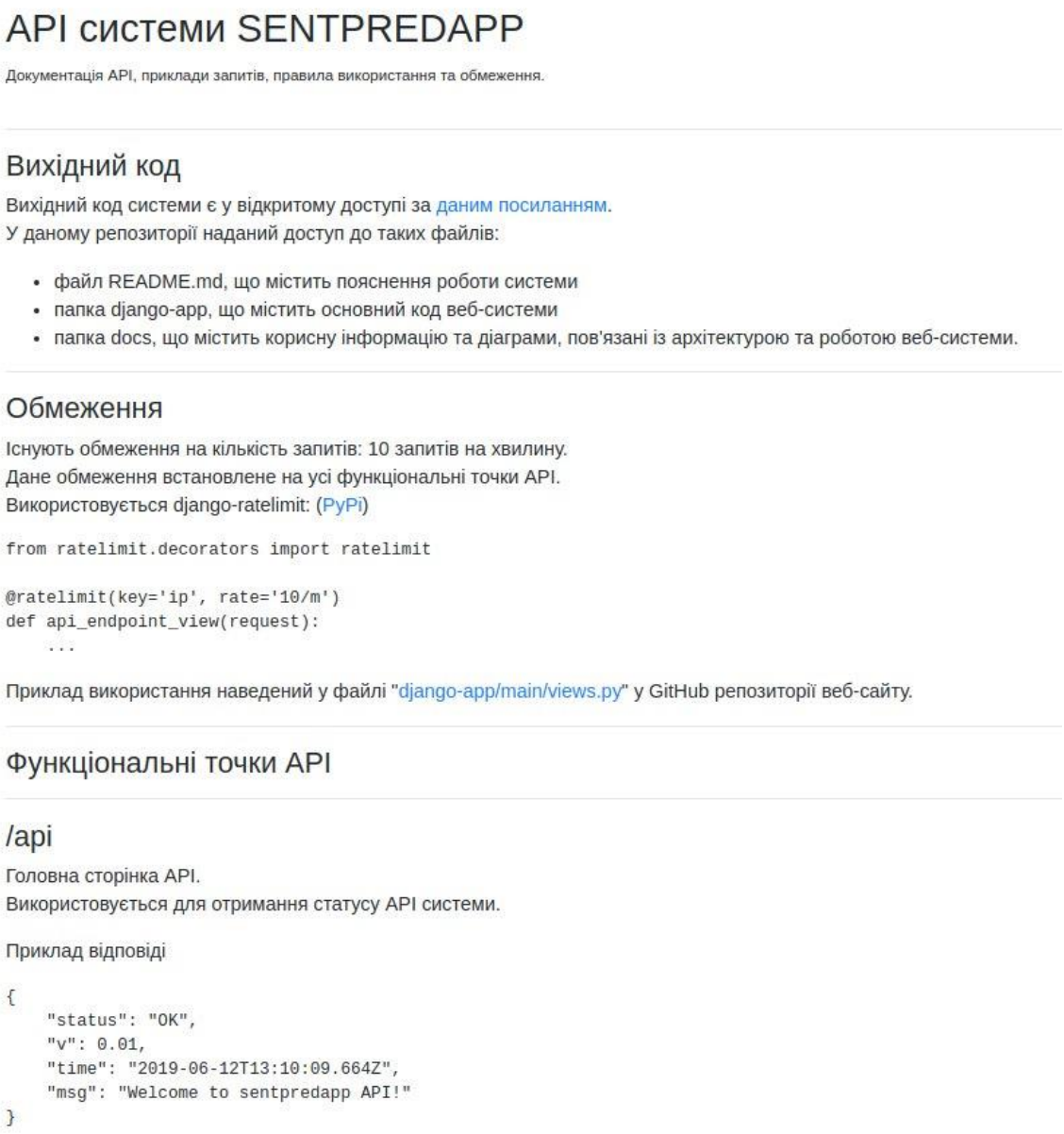


Рис. 4.9. Сторінка огляду API веб-додатку

Користувач системи перед переглядом будь-якої сторінки має погодитись із правилами використання сайту, а саме тим, що розробники системи не несуть відповідальності за помилки та упущення алгоритмів, а також тим, що дані та інформація на сайті не є фінансовою консультацією. Дана інформація зображена на модальному вікні, що затемняє усю область екрану, окрім вікна із правилами користуванням сайту. Знімок екрану вікна показаний на рис. 4.10. Якщо користувач не погодиться із зазначеними правилами, його буде перенаправлено на сторінку www.google.com, тим самим його буде обмежено у користуванні сайтом.

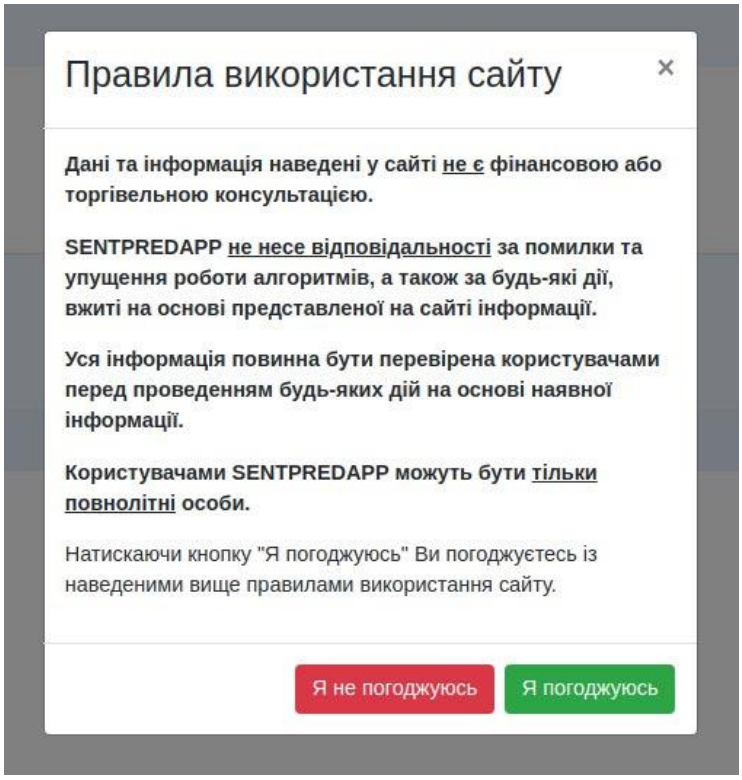


Рис. 4.10. Вікно правил використання сайту

4.3. Тестування алгоритму передбачення курсу

Існує багато способів визначення показників ефективності оцінки класифікатора, що аналізує полярність повідомлення, для того, щоб зрозуміти наскільки точно створена модель генералізує дані на нових, раніше невідомих, текстових даних. Одним із найчастіше використовуваних

підходів є метод перехресної перевірки (cross-validation method). Такий метод був застосований при розробці програмного забезпечення.

При тестуванні було взято очищені текстові дані у вигляді Dataframe, розміром 50,000 записів, розподілених на проміжку від 2019-01-01 по сьогоднішній день, а також усі фінансові дані за розглянутий проміжок.

Точність передбачень тональності текстів коректно реалізованими алгоритмами складає:

1. TextBlob: 74.17%.
2. VADER: 68.95%.
3. Наївний баєсів класифікатор: 71.64%.

Розглянуті алгоритми передбачення ціни криптовалюти показали наступні значення точності при валідації на тестовому, незалежному наборі даних:

1. Логістична регресія: 68.943%.
2. Поліноміальний баєсів класифікатор: 76.143%
3. Ансамблевий метод Random Forest 58.345%.

У порівнянні із існуючими аналогами розроблені алгоритми мають достатньо високий відсоток коректних передбачень, хоча і не найкращий із усіх розглянутих. Можливим покращенням роботи системи є реалізація наступних вдосконалень:

1. Використання трендів криптовалют від trends.google.com.
2. Використання метрик технічного аналізу на кшталт RSI.
3. Використання кількості заявок на продаж та купівлю певної криптовалюти за певний період часу.
4. Використання моделей, що використовують контекст слова у реченні при аналізі тональності текстових даних.

4.4. Рекомендації щодо подальшого використання

Для найкращого користування системою користувачу пропонується:

1. Повністю прочитати керівництво користувача.

2. Прочитати керівництво програміста, якщо користувач бажає використовувати функціональні точки API системи.
3. Користуватись оцінкою узагальненої ринкової капіталізації при передбаченні криптовалют, оскільки цей показник чітко пов’язаний зі зміною курсу Bitcoin, а отже і більшості інших криптовалют.
4. При виникненні питань зв’язуватись із розробником системи, контакти якого наведені на сторінці «Про систему».

ВИСНОВКИ

Метою даного дипломного проекту було розроблення програмної системи прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання.

Аналіз засобів розроблення такої програмної системи у вигляді веб-додатку, попередньо виконаний у дипломному проекті, показав доречність створення системи у її вигляді та доцільність використання вибраних алгоритмів передбачення курсу криптовалют та аналізу тональності тексту.

Розроблена програмна система:

1. Дозволяє отримувати передбачення курсу криптовалют Bitcoin, Ether та Litecoin.
2. Дозволяє оцінювати тональність більшості новин та повідомлень соціальних медіа Twitter та Reddit на предмет їх впливу на курс криптовалюти.
3. Має зрозумілий та зручний дизайн, розрахований на різні браузери та різноманітні пристрої із різною роздільністю екранів.
4. Містить API для надання стороннім розробникам доступу до результатів роботи системи.

Програмне забезпечення реалізоване у повному обсязі, усі функціональні та нефункціональні вимоги є виконаними, тестування продукту відтворено відповідно до затвердженої програми та методики тестування, проведена оцінка основних метрик використаних алгоритмів машинного навчання.

Використання розробленої системи забезпечить трейдерів та криптовалютних інвесторів автоматизованим інструментом, що допоможе уточнювати передбачення курсу криптовалют.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Keizer Söze. Bitcoin and Cryptocurrency Technologies: Bitcoin and cryptocurrency investing, cryptocurrency book for beginners [Текст] / Söze Keizer. – CreateSpace Independent Publishing Platform, 2017. – 506 с. – ISBN 197-906-068-1.
2. Aimee Vo. Cryptocurrency Trading & Investing: Beginners Guide To Trading & Investing In Bitcoin, Alt Coins & ICOs For Profit [Текст] / Aimee Vo. – CreateSpace Independent Publishing Platform, 2017. – 282 с. – ISBN 197-792-453-0.
3. Saifedean Ammous. The Bitcoin Standard: The Decentralized Alternative to Central Banking [Текст] / Ammous Saifedean. – 1st edition. – Wiley, 2018. – 304 с. – ISBN 111-947-386-1.
4. Coinmarketcap: Cryptocurrencies by Market Capitalization [Електронний ресурс]. – Режим доступу: https://coinmarketcap.com.
5. Technical Analysis [Електронний ресурс]. – 2011. – Режим доступу: https://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf.
6. Bakker Joris. Technical Analysis in the Cryptocurrency Market / Joris Bakker. – Erasmus School of Economics, 2017. – 53 с.
7. Andrius Mudinas, Market Trend Prediction using Sentiment Analysis: Lessons Learned and Paths Forward / Mudinas Andrius, Zhang Dell, Levene Mark. – Birkbeck: University of London, 2018. – 10 с.
8. Sarkis Agaian. Financial Sentiment Analysis Using Machine Learning Techniques / Agaian Sarkis, Kolm Petter. – New York: International Journal of Investment Management and Financial Innovations, 2017. – 9 с. – ISSN: 2381-1196.
9. Kelly Stephen. News, Sentiment, and Financial Markets: A Computational System to Evaluate the Influence of Text Sentiment on Financial Assets : дис. докт. філос. наук / Stephen Kelly. – Dublin: School of Computer Science and Statistics Trinity College Dublin, 2016. – 144 с.

10. Takuya Shintate. Trend Prediction Classification for High Frequency Bitcoin Time Series with Deep Learning / Shintate Takuya, Pichl Lukáš. – Tokyo: Journal of Risk and Financial Management, 2018. – 15 с.
11. Alessandretti Laura. Anticipating cryptocurrency prices using machine learning / Laura Alessandrettia, Abeer ElBahrawyb, Luca Maria Aielloc, Baronchelli Andrea. – 2019. – 24 с.
12. Lamon Connor. Cryptocurrency Price Prediction Using News and Social Media Sentiment / Connor Lamon, Eric Nielsen, Eric Redondo. – 6 с.
13. Kaminski Jermain. Nowcasting the Bitcoin Market with Twitter Signals / Jermain C. Kaminski. – MIT Media Lab, 2016. – 16 с.
14. Stenqvist Evita. Predicting Bitcoin price fluctuation with Twitter sentiment analysis / Evita Stenqvist, Jacob Lonno. – Stockholm, 2017. – 37 с.
15. Machine Learning [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Machine_learning.
16. Müller Andreas. Introduction to Machine Learning with Python: A Guide for Data Scientists [Текст] / Andreas Müller, Sarah Guido. – 1st edition. – O'Reilly Media, 2016. – 400 с. – ISBN: 144-936-941-3.
17. Géron Aurélien. Hands-On Machine Learning with Scikit-Learn and Tensorflow: Concepts, Tools and Techniques to Build Intelligent Systems [Текст] / Aurélien Géron. – 1st edition. – O'Reilly Media, 2017. – 574 с. – ISBN: 149-196-22-91.
18. Honchar Alexandr. Neural networks for algorithmic trading. Multivariate time series [Електронний ресурс] / Alexandr Honchar Ivanovich. – Режим доступу: https://medium.com/@alexrachnog/neural-networks-for-algorithmic-trading-2-1-multivariate-time-series-ab016ce70f57.
19. Palaniappan Vivek. Using Machine Learning to Predict Stock Prices [Електронний ресурс] / Vivek Palaniappan. – 2018. – Режим доступу: https://medium.com/analytics-vidhya/using-machine-learning-to-predict-stock-prices-c4d0b23b029a.

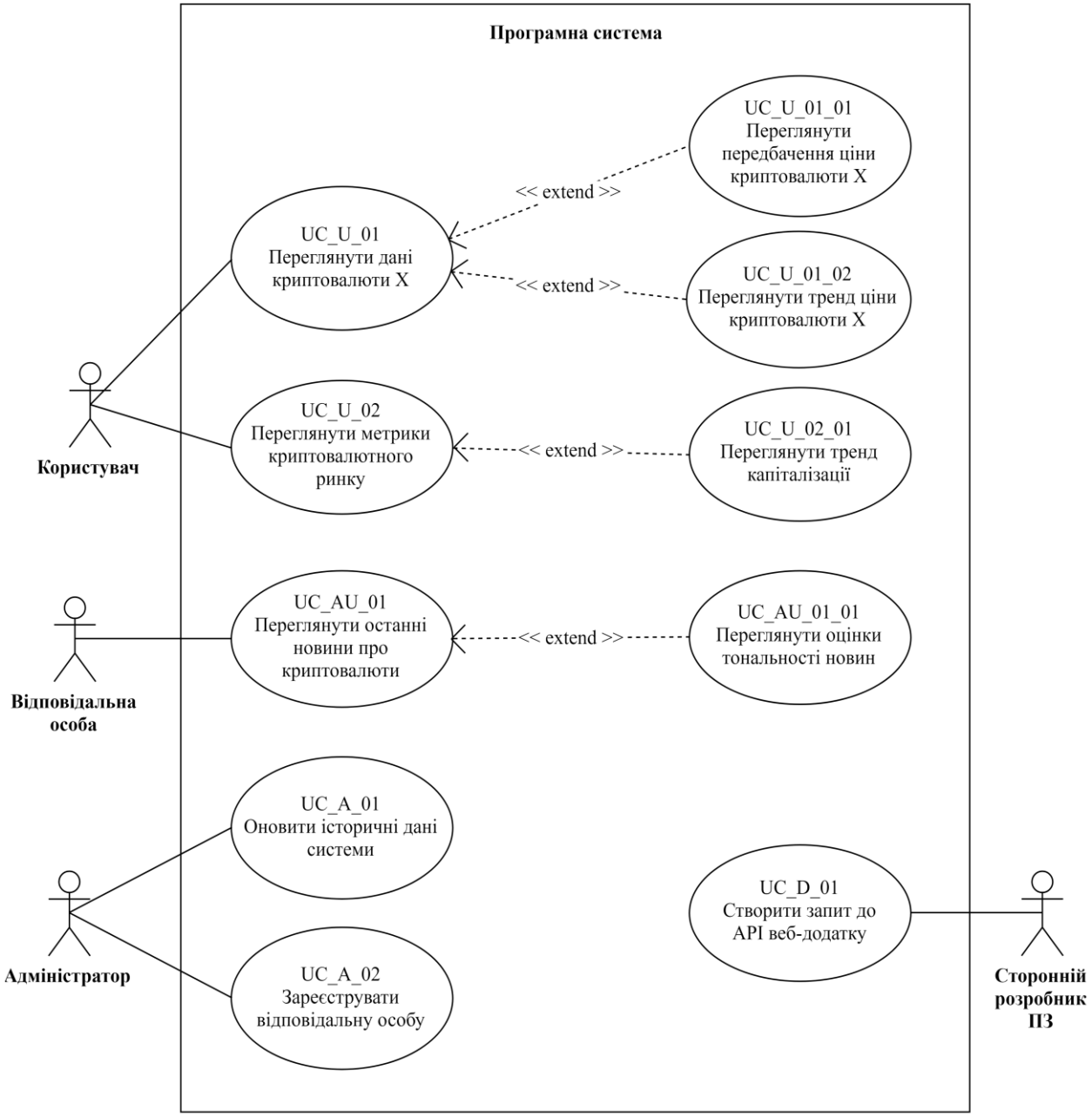
20. Coinpredictor – Cryptocurrency Price Predictions [Електронний ресурс]. – Режим доступу: https://coinpredictor.io.
21. Santiment – Crypto Trends in Social Media [Електронний ресурс]. – Режим доступу: https://app.santiment.net.
22. The TIE – The Most Powerful Tool for Crypto Traders [Електронний ресурс]. – Режим доступу: https://thetie.io.
23. Stock Market Prediction sentiment using analysis [Електронний ресурс]. – Режим доступу: http://bit.ly/2wISuAt.
24. Cryptocurrency Prediction Sentiment using Analysis [Електронний ресурс]. – Режим доступу: http:/bit.ly/2WoisUa.
25. Advantages of web applications over Desktop applications [Електронний ресурс]. – Режим доступу: http://bit.ly/2IpSRoR.
26. The State of Developer Ecosystem in 2018 [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/research/devecosystem-2018.
27. JavaScript in 2018 survey [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/research/devecosystem-2018/javascript.
28. Python in 2018 survey [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/research/python-developers-survey-2018.
29. Lindley Cody. Front-end Developer Handbook 2019 [Електронний ресурс] / Cody Lindley. – Режим доступу: http://bit.ly/2WqhC9H.
30. Sentiment Analysis: Nearly Everything You Need to Know [Електронний ресурс]. – Режим доступу: https://monkeylearn.com/sentiment-analysis.
31. Comparison of Top 6 Python NLP Libraries [Електронний ресурс]. – Режим доступу: https://www.kdnuggets.com/2018/07/comparison-top-6-python-nlp-libraries.html.
32. Simplifying Sentiment Analysis using VADER in Python on Social Media Text [Електронний ресурс]. – Режим доступу: http://bit.ly/31jmjW6.
33. VADER Sentiment Analysis vaderSentiment [Електронний ресурс]. – Режим доступу: https://github.com/cjhutto/vaderSentiment.

34. Description of the database normalization basics [Електронний ресурс]. – Режим доступу: http://bit.ly/2WvU7AL.
35. Working Structure of Django MTV Architecture [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/working-structure-of-django-mtv-architecture-a741c8c64082.

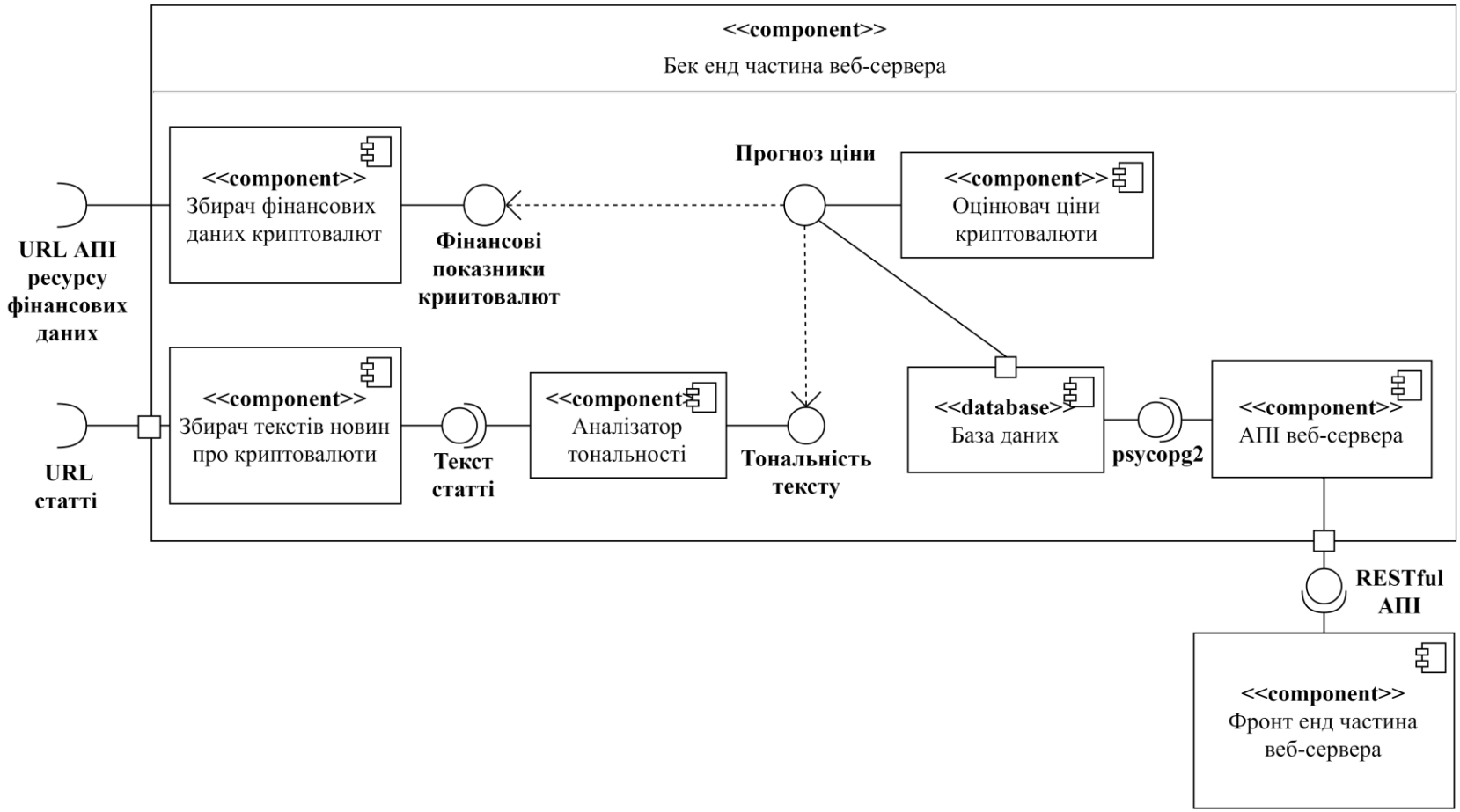
ДОДАТКИ

Додаток 1

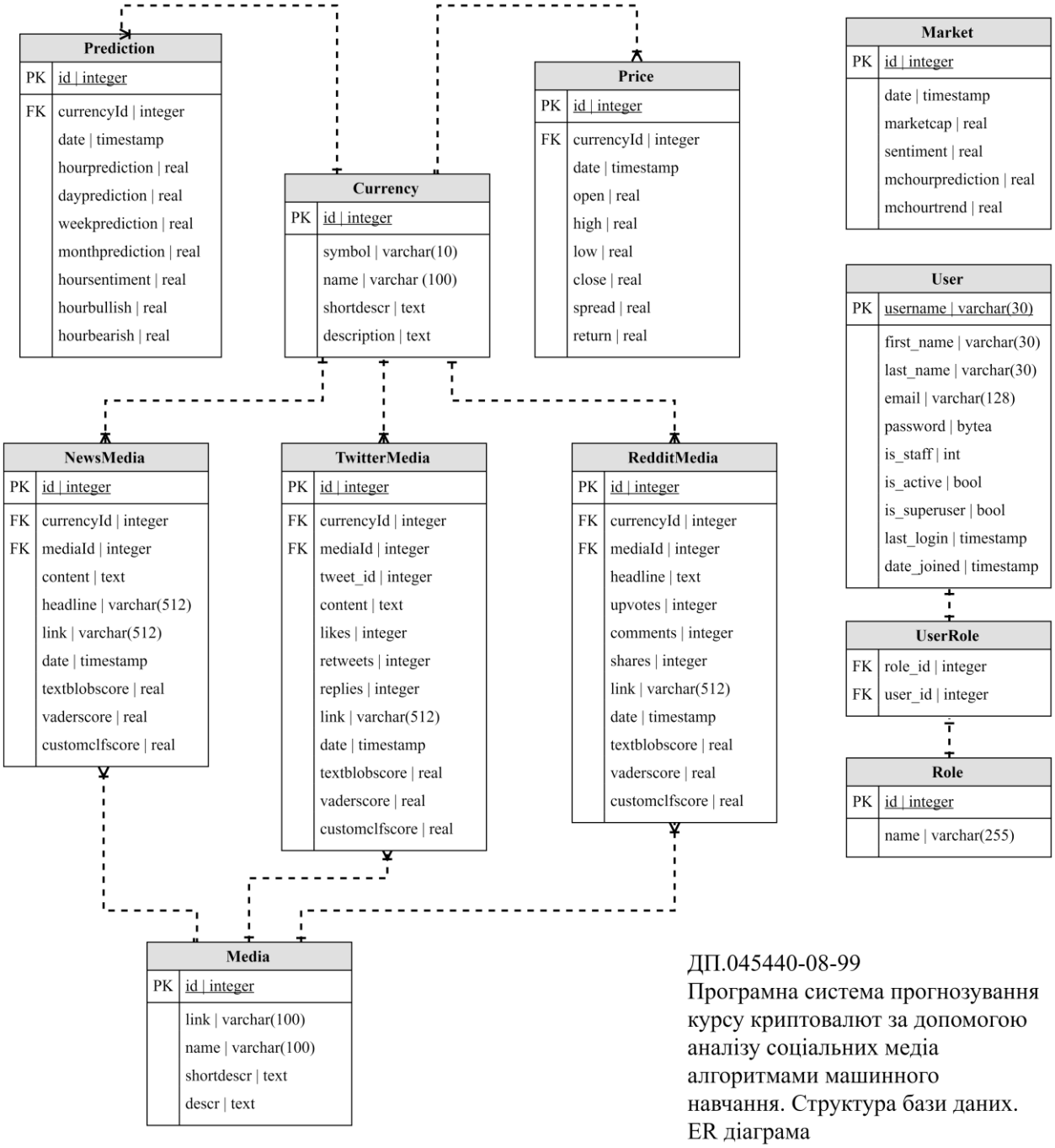
Копії графічних матеріалів



ДП.045440-06-99
Програмна система прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання. Функціональність програмних засобів. UML діаграма прецедентів

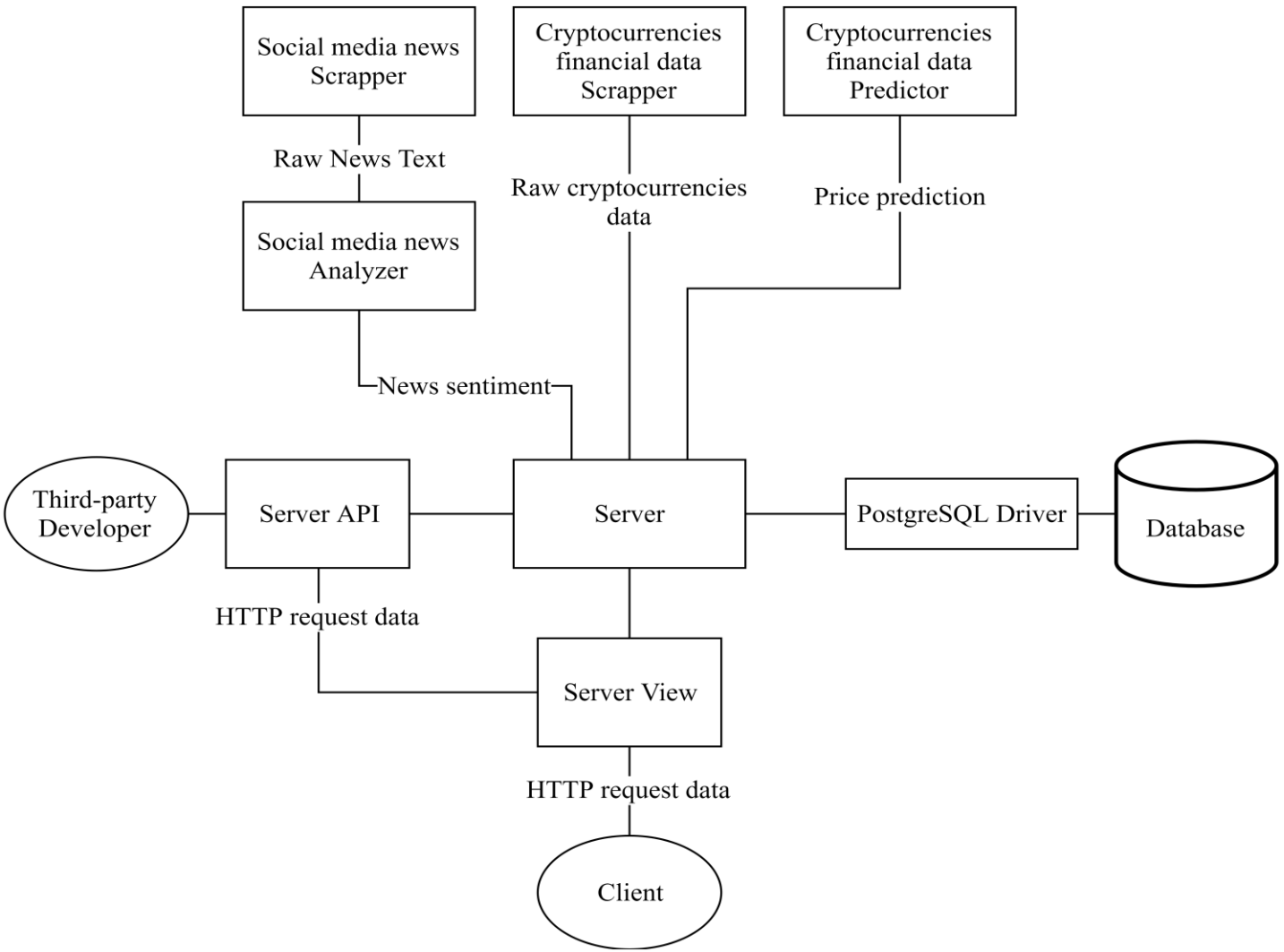


ДП.045440-07-99
Програмна система прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання. Компоненти та інтерфейси веб-додатку. UML діаграма компонентів



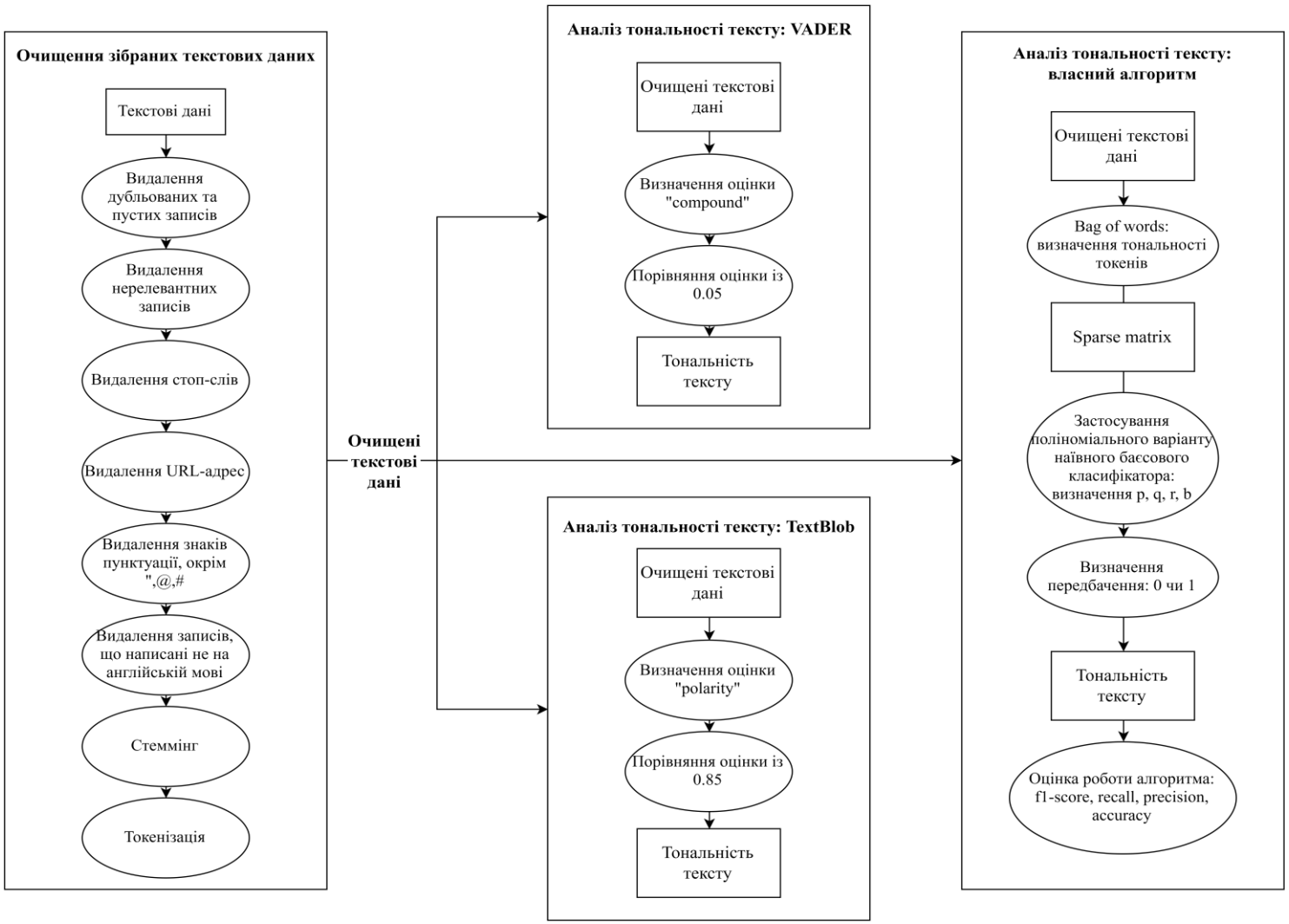
ДП.045440-08-99
Програмна система прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання. Структура бази даних. ER діаграма

Схема взаємодії програмних модулів



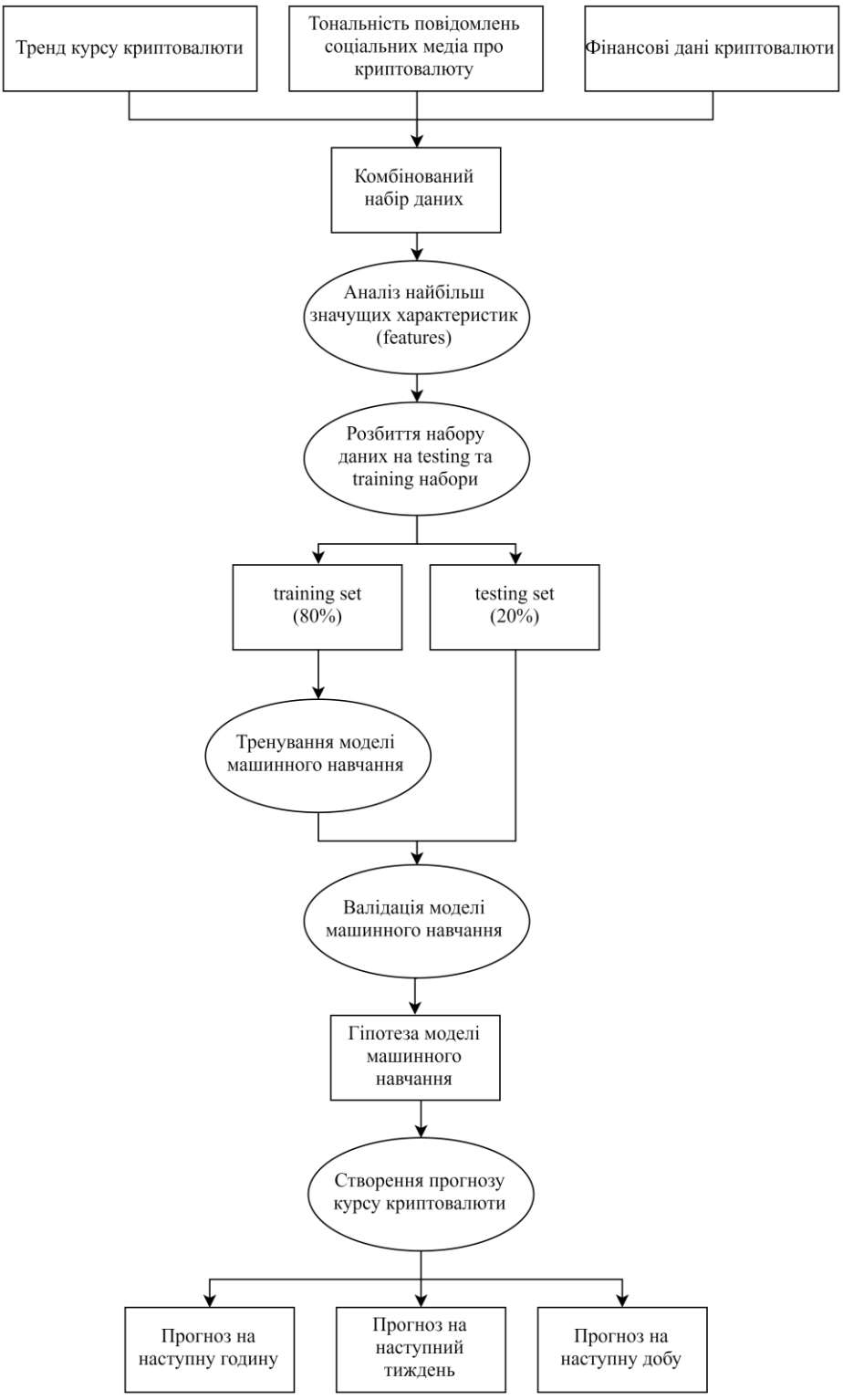
Гончар М.І., група КП-51

Алгоритм аналізу повідомлень соціальних медіа



Гончар М.І., група КП-51

Алгоритм прогнозування фінансових даних



Гончар М.І., група КП-51

Додаток 2

Лістинг модуля збору повідомлень соціальної мережі Twitter

```
import json
import random
import time
import urllib
from datetime import datetime, timedelta

import bs4
import psycopg2
import requests

# URLS to get Twitter search results
TWITTER_SEARCH_URL = 'https://twitter.com/search?q={0}&src=typd&qf=off&l=en'
TWITTER_SEARCH_MORE_URL =
'https://twitter.com/i/search/timeline?q={0}&src=typd&vertical=default&include
_available_features=1&include_entities=1&qf=off&l=en&max_position={1}'

# URLs to get Twitter user timeline.
TWITTER_USER_URL = 'https://twitter.com/{0}'
TWITTER_USER_MORE_URL =
'https://twitter.com/i/profiles/show/{0}/timeline/tweets?include_available_fea
tures=1&include_entities=1&max_position={1}'

# Different user-agent values to try to overcome bot protection.
user_agent_pool = [
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/74.0.3729.169 Safari/537.36',
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/58.0.3029.110 Safari/537.36',
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:53.0) Gecko/20100101
Firefox/53.0',
    'Mozilla/5.0 (compatible, MSIE 11, Windows NT 6.3; Trident/7.0; rv:11.0)
like Gecko',
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393'
]

# Different timeouts to try to overcome bot detection.
timeout_pool_s = [1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]

# Save IDs from database for Currency and Media models.
currency_id_by_ticker = {
    'BTC': 1,
    'ETH': 2,
    'LTC': 3,
    'GENERAL': 4
}
media_id_by_ticker = {
    'Twitter': 1,
    'Reddit': 2,
    'Cnbc': 3,
    'Forbes': 4,
    'Coindesk': 5,
    'Cointelegraph': 6,
    'Ccn': 7,
}
```

```
# Default values for sentiment, predictions and trends which identify
uninitialized data.
DEFAULT_SENT_SCORE = 100.0


class Tweet(object):
    """Representation of a single tweet from Twitter"""

    def __init__(self, **params):
        self.currencyId_id = params['currencyId_id']
        self.mediaId_id = params['mediaId_id']
        self.tweet_id = params['tweet_id']
        self.content = params['tweet_content']
        self.likes = params['favorite_cnt']
        self.retweets = params['retweet_cnt']
        self.replies = params['reply_cnt']
        self.link = params['tweet_link']
        self.date = params['timestamp']
        self.textblobscore = params['textblobscore']
        self.vaderscore = params['vaderscore']
        self.customclfscore = params['customclfscore']

    def jsonify(self):
        return {
            'currencyId_id': self.currencyId_id,
            'mediaId_id': self.mediaId_id,
            'tweet_id': self.tweet_id,
            'content': self.content,
            'likes': self.likes,
            'retweets': self.retweets,
            'replies': self.replies,
            'link': self.link,
            'date': self.date,
            'textblobscore': self.textblobscore,
            'vaderscore': self.vaderscore,
            'customclfscore': self.customclfscore,
        }

    def __repr__(self):
        return "Tweet {0}".format(" ".join([self.tweet_id,
self.date.strftime("%m/%d/%Y, %H:%M:%S")]))


class TwitterTimelineParser(object):
    """Class to hold tools for parsing Twitter timeline into a list of
main.scrape_twitter.Tweet instances"""

    def parse_tweets_timeline(self, timeline_html, currencyId_id):
        tweets = []
        soup = bs4.BeautifulSoup(timeline_html, "lxml")
        for tweet_tag in soup.find_all("div", class_="tweet"):
            # Find tweet ID.
            tweet_id = tweet_tag['data-tweet-id']
            # Find content of the tweet.
```

```
            tweet_content = tweet_tag.find('p', class_='tweet-text').text
            # Find emojis
            emojis_tags = tweet_tag.find('p', class_='tweet-
text').find_all(class_='Emoji')
            tweet_content += " " + " ".join(emoji_tag['alt'] for emoji_tag in
emojis_tags)
            # Find favorites, likes and replies cnt.
            tweet_tag_footer_div = tweet_tag.find('div', class_='stream-item-
footer')
            favorite_cnt = self.find_tweet_cnt_stats(tweet_tag_footer_div,
'favorite')
            retweet_cnt = self.find_tweet_cnt_stats(tweet_tag_footer_div,
'retweet')
            reply_cnt = self.find_tweet_cnt_stats(tweet_tag_footer_div,
'reply')
            # Find tweet link.
            tweet_link = 'twitter.com{0}'.format(tweet_tag['data-permalink-
path'])
            # Find tweet posting timestamp.
            timestamp_unixlike = tweet_tag.find('span',
class_='_timestamp')['data-time']
            timestamp = datetime.utcfromtimestamp(int(timestamp_unixlike))
            # Create Tweet class instance to save.
            tweet_instance = Tweet(
                currencyId_id=currencyId_id,
                mediaId_id=media_id_by_ticker['Twitter'],
                tweet_id=tweet_id,
                tweet_content=tweet_content,
                favorite_cnt=favorite_cnt, retweet_cnt=retweet_cnt,
reply_cnt=reply_cnt,
                tweet_link=tweet_link,
                timestamp=timestamp,
                textblobscore=DEFAULT_SENT_SCORE,
                vaderscore=DEFAULT_SENT_SCORE,
                customclfscore=DEFAULT_SENT_SCORE
            )
            # Save Tweet class instance.
            tweets.append(tweet_instance)
        return tweets

    @staticmethod
    def find_tweet_cnt_stats(tweet_footer_html, stats_type):
        if stats_type not in ['reply', 'retweet', 'favorite']:
            raise ValueError('Incorrect stats_type value')
        stats_span = tweet_footer_html.find(
            'span', class_='ProfileTweet-action--{0}'.format(stats_type))
        stats_span_value = stats_span.find(
            'span', class_='ProfileTweet-actionCount')['data-tweet-stat-
count']
        return stats_span_value


class TwitterScraper(object):
    """Implementation of scraping Twitter search page and user page timelines
```

```
    scrape_search_timeline - use it for hashtag and keyword searching
        scrapes ~500 tweets

    scrape_userpage_timeline_adv_search - use it when you know the username
        but want only a portion of it's tweets filtered by date.
        Works poorly for long-term search.
        Use it only for 1-2 day range.
        scrapes ~120 tweets for realdonaldtrump and ~50 for WhaleDump

    scrape_userpage_timeline - best for getting last tweets from user.
        scrapes ~700-800 tweets
    """

    def __init__(self):
        self._timeline_parser = TwitterTimelineParser()

    def scrape_search_timeline(self, search_query, currencyId_id):
        # Build a search query.
        search_query = "{0}".format(search_query)
        search_query = urllib.parse.quote(search_query)
        # Start scraping search results page timeline.
        for tweet_bunch in self._scrape_timeline(TWITTER_SEARCH_URL,
TWITTER_SEARCH_MORE_URL, search_query, currencyId_id):
            yield tweet_bunch

    def scrape_search_timeline_adv_search(self, search_query, from_datetime,
to_datetime, currencyId_id):
        # Build a search query.
        search_query = "{0} since:{1} until:{2}".format(
            search_query,
            from_datetime.strftime("%Y-%m-%d"), to_datetime.strftime("%Y-%m-
%d")
        )
        search_query = urllib.parse.quote(search_query)
        # Start scraping search results page timeline.
        is_keyword_search = True
        for tweet_bunch in self._scrape_timeline(TWITTER_SEARCH_URL,
TWITTER_SEARCH_MORE_URL, search_query, currencyId_id, is_keyword_search):
            yield tweet_bunch

    def scrape_userpage_timeline(self, username):
        """Please note, that this method could scrape only around 800 last
tweets from user page
        This method should be used as a faster approach to scrape selected
users's tweets.
        """
        # Start scraping user page timeline.
        currencyId_id = currency_id_by_ticker['GENERAL']
        for tweet_bunch in self._scrape_timeline(TWITTER_USER_URL,
TWITTER_USER_MORE_URL, username, currencyId_id):
            yield tweet_bunch

    def scrape_userpage_timeline_adv_search(self, from_username,
from_datetime, to_datetime):
        # Build a search query.
```

```
        search_query = "from:{0} since:{1} until:{2}".format(
            from_username,
            from_datetime.strftime("%Y-%m-%d"), to_datetime.strftime("%Y-%m-
%d")
        )
        search_query = urllib.parse.quote(search_query)
        # Start scraping user's tweets by using advanced search.
        currencyId_id = currency_id_by_ticker['GENERAL']
        adv_search = True
        for tweet_bunch in self._scrape_timeline(TWITTER_SEARCH_URL,
TWITTER_SEARCH_MORE_URL, search_query, currencyId_id, adv_search):
            yield tweet_bunch

    def _scrape_timeline(self, first_page_url, more_page_url, main_term,
currencyId_id, adv_search=False, is_keyword_search=False):
        # Perform scraping on the first page.
        first_page_tweets_bunch, next_position =
self._scrape_first_page(first_page_url, main_term, currencyId_id)
        yield first_page_tweets_bunch
        # Perform scraping on the other pages.
        for nth_page_tweets_bunch in self._scrape_more_pages(more_page_url,
main_term, next_position, adv_search, currencyId_id, is_keyword_search):
            yield nth_page_tweets_bunch

    def _scrape_first_page(self, first_page_url, main_term, currencyId_id):
        # Request timeline's 1st page data.
        request_url = first_page_url.format(main_term)
        USER_AGENT = random.choice(user_agent_pool)
        response = requests.get(request_url, headers={'User-agent':
USER_AGENT})
        response_text = response.text
        # Scrape tweets from the 1st page.
        first_page_parsed_tweets =
self._timeline_parser.parse_tweets_timeline(response_text, currencyId_id)
        # Find next position argument for the next timeline page.
        next_position = self._find_arg_value(response_text, "data-min-
position")
        # Mandatory sleep.
        time.sleep(timeout_pool_s[0])
        return first_page_parsed_tweets, next_position

    def _scrape_more_pages(self, more_page_url, main_term, next_position,
adv_search, currencyId_id, is_keyword_search):
        # Vars-helpers for the method
        old_next_position = None
        has_more_items = True  # because bool(next_position) = True
        # Loop while more pages available. Any issue - request data again with
the same next_position param.
        while has_more_items:
            try:
                # Request timeline data from Nth page.
                request_url = more_page_url.format(main_term, next_position)
                USER_AGENT = random.choice(user_agent_pool)
                response = requests.get(request_url, headers={'User-agent':
USER_AGENT})
```

```
                response_text = response.text
                response_dict = json.loads(response_text)
                # Scrape tweets from the Nth page.
                nth_page_parsed_tweets =
self._timeline_parser.parse_tweets_timeline(response_dict['items_html'],
currencyId_id)
                # Get next_position value for the N+1th page.
                next_position = response_dict.get('min_position', None)
                yield nth_page_parsed_tweets
                # Check if N+1th page exists. If not - exit the function.
                has_more_items = old_next_position != next_position
                if is_keyword_search:
                    if not has_more_items:
                        break
                if not response_dict['has_more_items']:
                    if not has_more_items:
                        break
                    if adv_search:
                        break
                # Save old_next_position for the next N+1th page existence
check.
                old_next_position = next_position
                # Sleep for 1-5s before the next request.
                sleep_time = random.choice(timeout_pool_s)
                time.sleep(sleep_time)
            except Exception:
                break

    @staticmethod
    def _find_arg_value(html, value):
        start_pos = html.find(value) + len(value)
        start_pos += 2  # skip = and " characters.
        end_pos = html.find('"', start_pos)
        return html[start_pos:end_pos]


class TwitterScraperPerformer(object):

    def __init__(self):
        # DB connection and cursor instances.
        self.conn = psycopg2.connect()
        self.cur = self.conn.cursor()
        # Saved list of tweets IDs from DB.
        self.existing_tweets_ids = self._get_existing_tweets_ids()
        # Twitter scrapper instance.
        self.twitter_scraper = TwitterScraper()
        # Selected Twitter usernames and hashtags to scrape.
        self.usernames = [
            'VitalikButerin', 'SatoshiLite', 'aantonop', 'ErikVoorhees',
'brian_armstrong', 'cz_binance', 'saifedean',  # influencers
            'binance', 'bitmexdotcom', 'bitfinex', 'coinbase', 'cex_io',
'krakenfx',  # cryptocurrency exchanges
            'bitmexresearch', 'wizsecurity', 'proofofresearch',  # research
pages
```

```
            'VentureCoinist', 'notsofast', 'Cryptopia_NZ', 'BTC_Revolution',
'FelixOHartmann', 'CryptoMoshing', 'cryptotraderpro',  # cryptocurrency
traders
            'bitcoin', 'ethereum', 'litecoin',  # "official" cryptocurrency
accounts
        ]
        # Initialize hashtags, tags and keywords to scrape
        self.ticker_hashtags_tags = {
            'BTC': ["#btc", "#Bitcoin", "$btc"],
            'ETH': ["#eth", "#Ethereum", "$eth"],
            'LTC': ['#ltc', "#Litecoin", "$ltc"],
            'GENERAL': ['#cryptocurrency', "#crypto"]
        }
        # The earliest date to scrape.
        self.earliest_date = datetime(2016, 1, 1)

    def scrape_last_updated_data(self):
        # Update local storage of already saved tweets.
        self.existing_tweets_ids = self._get_existing_tweets_ids()
        # Container to save new scraped data.
        data_container = []
        # Perform scraping by date range.
        # 1. Perform scraping by username.
        for username in self.usernames:
            for tweet_bunch in
self.twitter_scraper.scrape_userpage_timeline(username):
                skip_user = False
                for tweet in tweet_bunch:
                    if tweet.tweet_id not in self.existing_tweets_ids:
                        self._save_tweet(tweet)
                        data_container.append(tweet)
                    else:
                        skip_user = True
                        break
                if skip_user is True:
                    print('DBG: skip user {0}\n'.format(username))
                    break
        # Update local storage of existing tweets ids.
        self.existing_tweets_ids = self._get_existing_tweets_ids()
        # 2. Perform scraping by keywords.
        for ticker, search_terms in self.ticker_hashtags_tags.items():
            currencyId_id = currency_id_by_ticker[ticker]
            for search_term in search_terms:
                for tweet_bunch in
self.twitter_scraper.scrape_search_timeline(search_term, currencyId_id):
                    skip_search_term = False
                    for tweet in tweet_bunch:
                        if tweet.tweet_id not in self.existing_tweets_ids:
                            self._save_tweet(tweet)
                            data_container.append(tweet)
                        else:
                            skip_search_term = True
                            break
                    if skip_search_term is True:
```

```
                        print('DBG: skip search term
{0}\n'.format(search_term))
                        break
        # Return data and status for webpage view.
        return {
            'status': 'OK',
            'data': data_container
        }

    def scrape_date_range_data(self, from_date, to_date):
        # Check if from_date is no older than the earliest date allowed.
        if from_date < self.earliest_date:
            from_date = self.earliest_date
        # Update local storage of already saved tweets.
        self.existing_tweets_ids = self._get_existing_tweets_ids()
        # Container to save new scraped data.
        data_container = []
        # Perform scraping by date range.
        # 1. Perform scraping by username.
        for username in self.usernames:
            print('Start scraping {0}\n'.format(username))
            for tweet_bunch in
self.twitter_scraper.scrape_userpage_timeline_adv_search(username, from_date,
to_date):
                print('Go to next bunch for {0}\n'.format(username))
                for tweet in tweet_bunch:
                    if tweet.tweet_id not in self.existing_tweets_ids:
                        self._save_tweet(tweet)
                        data_container.append(tweet)
                    else:
                        print('continue {0}\n'.format(username))
                        continue
        # Update local storage of existing tweets ids.
        self.existing_tweets_ids = self._get_existing_tweets_ids()
        # 2. Perform scraping by keywords.
        for ticker, search_terms in self.ticker_hashtags_tags.items():
            currencyId_id = currency_id_by_ticker[ticker]
            for search_term in search_terms:
                print('Start scraping {0}\n'.format(search_term))
                for tweet_bunch in
self.twitter_scraper.scrape_search_timeline_adv_search(search_term, from_date,
to_date, currencyId_id):
                    print('Go to next bunch for {0}\n'.format(search_term))
                    for tweet in tweet_bunch:
                        if tweet.tweet_id not in self.existing_tweets_ids:
                            self._save_tweet(tweet)
                            data_container.append(tweet)
                        else:
                            print('continue {0}\n'.format(search_term))
                            continue
        # Return data and status for webpage view.
        return {
            'status': 'OK',
            'data': data_container
        }
```

```
    def get_last_tweets(self, n_tweets):
        select_query = """SELECT * FROM main_twittermedia ORDER BY date DESC
LIMIT %s;"""
        select_query_fields = (n_tweets, )
        self.cur.execute(select_query, select_query_fields)
        tweets_data_from_db = self.cur.fetchall()
        tweets = [self._tweet_jsonify_from_db(tweet) for tweet in
tweets_data_from_db]
        return tweets

    @staticmethod
    def _tweet_jsonify_from_db(row):
        return Tweet(
            tweet_content=row[1],
            favorite_cnt=row[2],
            retweet_cnt=row[3],
            reply_cnt=row[4],
            tweet_link=row[5],
            timestamp=row[6],
            textblobscore=row[7],
            vaderscore=row[8],
            customclfscore=row[9],
            currencyId_id=row[10],
            mediaId_id=row[11],
            tweet_id=row[12]
        )

    def _get_existing_tweets_ids(self):
        # Save Tweet IDs
        self.cur.execute("""select tweet_id from main_twittermedia;""")
        existing_tweets_fromdb = self.cur.fetchall()
        existing_tweets_ids = [row[0] for row in existing_tweets_fromdb]
        return existing_tweets_ids

    def _save_tweet(self, tweet):
        insert_query = """
            INSERT INTO main_twittermedia
            ("mediaId_id", "currencyId_id", tweet_id, content, likes,
retweets, replies, link, date, textblobscore, vaderscore, customclfscore)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);"""
        insert_query_fields = (
            tweet.mediaId_id, tweet.currencyId_id,
            tweet.tweet_id,
            tweet.content,
            tweet.likes, tweet.retweets, tweet.replies,
            tweet.link,
            tweet.date,
            tweet.textblobscore, tweet.vaderscore, tweet.customclfscore
        )
        self.cur.execute(insert_query, insert_query_fields)
        self.conn.commit()
        print('DBG INSERT: inserted {0}'.format(tweet))
```

```
def __dbg():
    ts = TwitterScraper()
    from_date = datetime(2018, 5, 29)
    to_date = datetime(2018, 5, 30)
    general_currencyId_id = currency_id_by_ticker['GENERAL']
    btc_currencyId_id = currency_id_by_ticker['BTC']
    tweets_container = []
    # for tweet_bunch in ts.scrape_search_timeline('#bitcoin',
btc_currencyId_id):
    # for tweet_bunch in ts.scrape_search_timeline_adv_search("#bitcoin",
from_date, to_date, btc_currencyId_id):
    # for tweet_bunch in ts.scrape_userpage_timeline_adv_search('WhalePanda',
from_date, to_date):
    for tweet_bunch in ts.scrape_userpage_timeline('notsofast'):
        tweets_container.extend(tweet_bunch)


def __dbg_db():
    from_date = datetime(2018, 5, 29)
    to_date = datetime(2018, 5, 30)
    tsp = TwitterScraperPerformer()
    resp = tsp.scrape_last_updated_data()
    resp = tsp.scrape_date_range_data(from_date, to_date)


def __dbg_get():
    tsp = TwitterScraperPerformer()
    data = tsp.get_last_tweets(20)
    print(data)


def __dbg_truncate_db():
    try:
        conn = psycopg2.connect()
        cur = conn.cursor()
        cur.execute("""TRUNCATE TABLE  main_twittermedia;""")
        conn.commit()
        print('Successfully truncated main_twittermedia')
    except (Exception, psycopg2.Error) as e:
        print('Couldn\'t truncate main_twittermedia table. Error:
{0}'.format(e))


def __historical_big_date_range():
    tsp = TwitterScraperPerformer()
    date_from = datetime(2019, 1, 1)
    date_to = datetime.now()
    while date_from < date_to:
        date_from_plus_1d = date_from + timedelta(days=1)
        tsp.scrape_date_range_data(date_from, date_from_plus_1d)
        date_from = date_from_plus_1d

@ratelimit(key='ip', rate='10/m')
@csrf_exempt
def api_update_range_data(req):
    resp_data = {}
    if req.method == 'POST':
        post_req_dict = req.POST
        print('\n{0}\n'.format(post_req_dict))
```

```
        if not post_req_dict['btn_id'] or not post_req_dict['from_time'] or
not post_req_dict['to_time']:
            resp_data['status'] = 'FAIL'
            return JsonResponse(resp_data)
        from_date = datetime.strptime(post_req_dict['from_time'], "%m/%d/%Y")
        to_date = datetime.strptime(post_req_dict['to_time'], "%m/%d/%Y")
        button_id = post_req_dict['btn_id']
        if button_id == "updTwitterRange":
            resp_data =
twitter_scraper_performer.scrape_date_range_data(from_date, to_date)
            resp_data_jsonified = [tweet.jsonify() for tweet in
resp_data['data']]
            resp_data['data'] = resp_data_jsonified
            # todo: clean twitter data
        elif button_id == "updRedditRange":
            pass
        elif button_id == "updNewsRange":
            pass
        else:
            resp_data['status'] = 'FAIL'
    else:
        resp_data['status'] = 'FAIL'
    return JsonResponse(resp_data)
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНА СИСТЕМА ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ ЗА ДОПОМОГОЮ АНАЛІЗУ СОЦІАЛЬНИХ МЕДІА АЛГОРИТМАМИ МАШИННОГО НАВЧАННЯ

Виконав: студент групи КП-51 Гончар Максим Іванович

Науковий керівник: ст. викл. Гадиняк Руслан Анатолійович

Київ – 2019

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити програмну систему, що надає передбачення курсу криптовалют шляхом аналізу тональності повідомлень соціальних медіа та застосування алгоритмів машинного навчання до зібраних і проаналізованих даних.

Завдання:

1. Проаналізувати галузь застосування та існуючі аналоги.
2. Розробити програмне забезпечення для створення прогнозів ціни криптовалют.
3. Протестувати розроблений алгоритм передбачення та можливості веб-додатку.

АКТУАЛЬНІСТЬ

1. Інструмент для учасників торгів:
   - збереження часу на аналіз ринку;
   - зменшення ризиків на покупку-продаж активів;
   - збільшення прибутків за рахунок покращеного розуміння ринку.
2. Застосування розробленого підходу до традиційних активів або акцій фондового ринку.

АНАЛІЗ НАЯВНИХ РІШЕНЬ

- **Аналоги**:
  - coinpredictor.io;
  - santiment.net;
  - thetie.io.

- **Недоліки аналогів**:
  - закриті системи;
  - точність – 75%;
  - одне джерело новин: Twitter.

ЗАСОБИ РЕАЛІЗАЦІЇ

- Мова програмування **Python**.
- Реляційна база даних **PostgreSQL** та драйвер **psycopg2**.
- Веб-фреймворк **Django.**
- Бібліотека **pandas** для збереження та маніпуляції даними.
- **BeautifulSoup (bs4)** для збору даних.
- Планувальних задач **Cron**.
- Бібліотеки **VADER** та **TextBlob** для аналізу тональності текстових даних.
- Бібліотека **scikit-learn** – реалізація математичних моделей для передбачення значень часового ряду.

БАЗА ДАНИХ

- Реляційна база даних PostgreSQL.
- Проведено першу, другу та третю нормалізації.
- Індекси для швидшого пошуку даних.

- Основні таблиці:
  - фінансові дані: Price, Prediction, Market;
  - текстові дані: NewsMedia, TwitterMedia, RedditMedia;
  - загальні інформаційні дані: Media, Market;
  - дані користувача: User, UserRole, Role.

АРХІТЕКТУРА ТА КОМПОНЕНТИ СИСТЕМИ

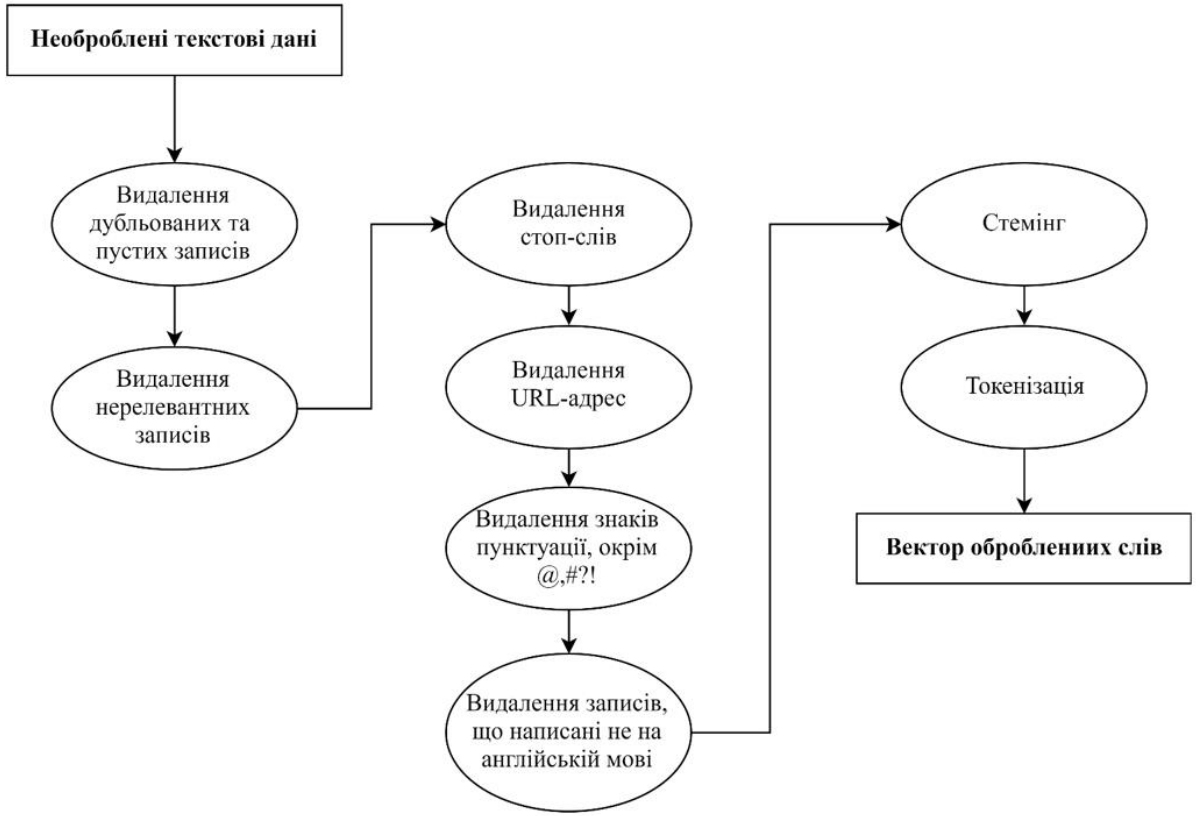
- Типи модулів:
  - модулі збору даних;
  - модулі оброблення та аналізу зібраних даних;
  - модулі візуалізації роботи системи.

- Користувачі:
  - клієнт;
  - сторонній розробник ПЗ.

РОЛІ КОРИСТУВАЧІВ СИСТЕМИ

1. Переглядач
   - перегляд передбачень цін криптовалют;
   - перегляд передбачень загальної ринкової капіталізації;
   - перегляд загальних даних ринку та досліджуваних криптовалют.
2. Довірена особа
   - ручна зміна тональностей зібраних текстів.
3. Адміністратор
   - ручне оновлення історичних фінансових та текстових даних;
   - реєстрація довірених осіб.

ОБРОБЛЕННЯ ЗІБРАНИХ ТЕКСТОВИХ ДАНИХ



АЛГОРИТМ АНАЛІЗУ ТОНАЛЬНОСТІ

1. TextBlob. Оцінки «neg», «pos». Інтервал [–1; 1].
2. VADER. Оцінка «compound». Інтервал [–1; 1].
3. Власний алгоритм класифікації:
   - метод «Мішок слів»;
   - наївний баєсів класифікатор;
   - дискретні величини: {–1; +1}.

- Точність:
  - TextBlob: 79.17%;
  - VADER: 73.95%;
  - наївний баєсів класифікатор: 76.64%.

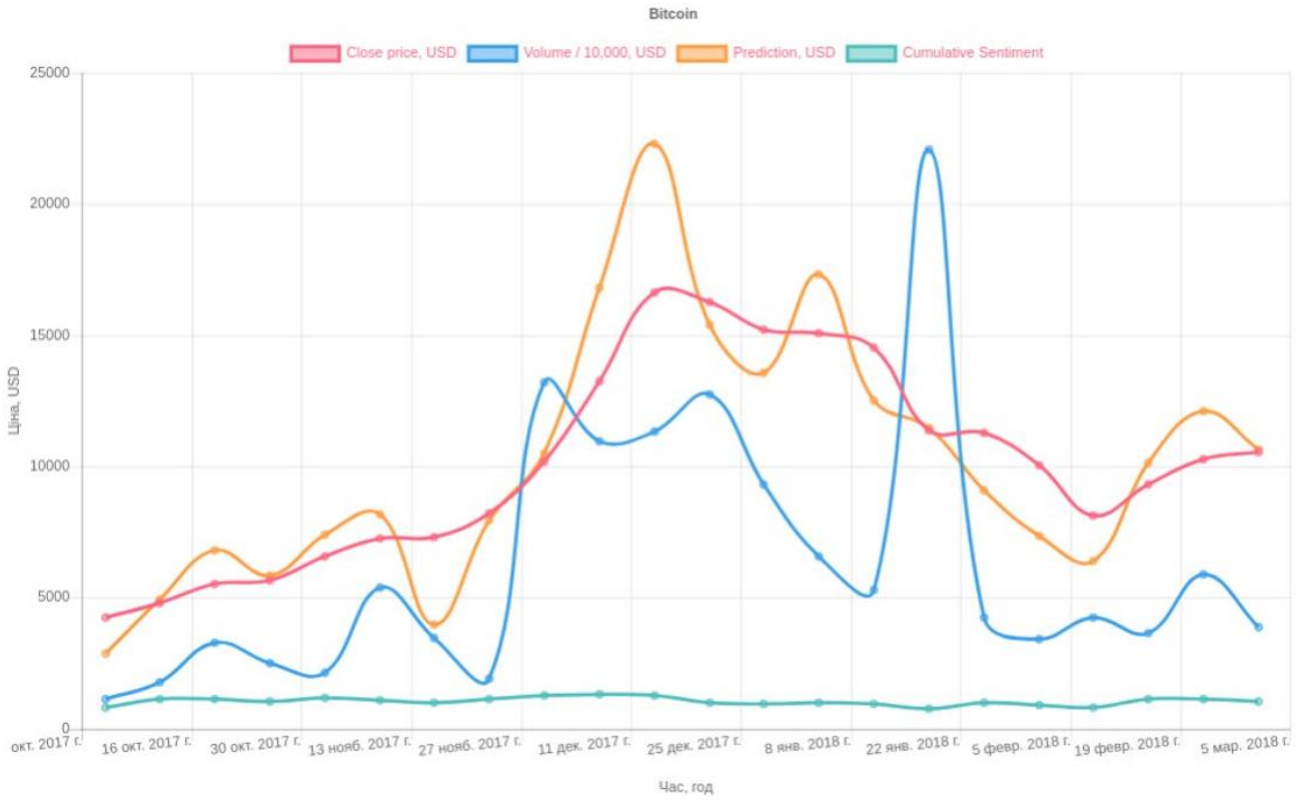
СТВОРЕННЯ ПЕРЕДБАЧЕННЯ ЦІНИ

1. Збір набору даних: комбінація фінансових даних та оцінок тональностей.
2. Розділення набору даних на набори для тренування (80%) та тестовий набір (20%).
3. Застосування набору для тренування математичної моделі та створення гіпотези.
4. Валідація отриманих результатів, використовуючи тестовий набір даних.
5. Подальший аналіз результатів роботи системи, уточнення параметрів моделей.

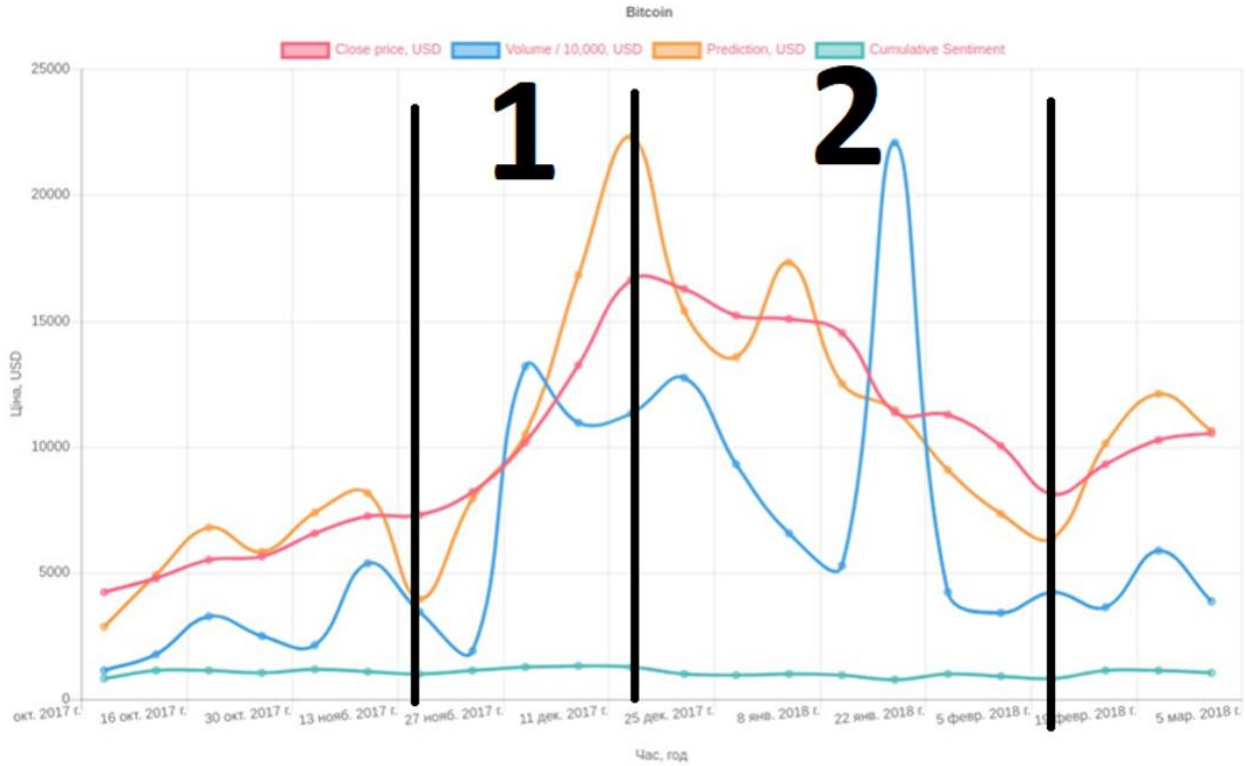
ОЦІНКА ТОЧНОСТІ АЛГОРИТМУ ПЕРЕДБАЧЕННЯ

- Передбачення ціни, USD:
  - Логістична регресія: 78.943%.
  - Ансамблевий метод Random Forest: 68.345%.
- Передбачення тренду, {–1; +1}:
  - Поліноміальний баєсів класифікатор: 86.143%.

ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ



ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ



ОЦІНКА РОЗРОБЛЕНОЇ СИСТЕМИ

- Масштабованість системи:
  - імплементовані шаблони проектування «Будівельник», «Адаптер» та «Декоратор».
- Безпека:
  - застосування CSRF токенів для форм вводу даних;
  - ліміт кількості запитів до функціональних точок API;
  - безпечний формат запитів до бази даних;
  - обмеження на логін та пароль нових довірених осіб.
- Надійність
  - збір даних проводиться із зміною IP, параметрів user-agent та різною затримкою;
  - точність передбачення текстів та ціни достатня.
- Швидкодія
  - швидке відкриття сторінок: менше ніж 0.5 секунди;
  - збір текстових даних за один день триває менше ніж 2 хвилини;
  - збір фінансових даних за увесь період триває менше ніж 20 секунд.

ВИСНОВКИ

Побудована система, що складється з:

- Модулів збору фінансових та текстових даних.
- Модулю аналізу тональності текстів.
- Модулю створення передбачень курсу криптовалют.
- Веб-додатку, що візуалізує результати роботи.

Порівняні точності визначення тональності текстів різними підходами.

Порівняні точності визначення передбачення ціни цифрових валют різними алгоритмами машинного навчання.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А.Дичка

"___" _____ 2018 р.

ПРОГРАМНА СИСТЕМА ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ ЗА ДОПОМОГОЮ АНАЛІЗУ СОЦІАЛЬНИХ МЕДІА АЛГОРИТМАМИ МАШИННОГО НАВЧАННЯ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ М.І. Гончар

2018

ЗМІСТ

1. ОБ'ЄКТ ВИПРОБУВАНЬ

Програмна система прогнозування курсу криптовалют за допомогою аналізу соціальних медіа алгоритмами машинного навчання. Дана програмна система імплементована у вигляді веб-додатка, що написаний мовою програмування Python 3 із використанням веб-фреймворку Django.

2. МЕТА ТЕСТУВАННЯ

Метою тестування є перевірка наступних елементів:

1. Доступ до бази даних через основну серверну частину.
2. Робота сервера із модулем, що відповідає за аналіз тональності текстових даних.
3. Робота сервера із модулем, що відповідає за створення передбачень ціни криптовалют.
4. Обробка запитів сторонніх користувачів до відкритого API.
5. Робота функціональних елементів сторінок веб-сторінки.
6. Забезпечення достатньої безпеки даних.
7. Зручність користування веб-додатком.
8. Відповідність дизайну вимогам програмного забезпечення.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Даний метод, що також називається Gray Box Analysis, використовується у тому сценарії, коли тестувальник має обмежені знання про внутрішні деталі системи. У такому тестуванні перевіряється як вихідний код, так і сам програмний продукт та відповідність існуючим програмним функціональним та нефункціональним вимогам. Gray Box Testing відрізняється від White Box

Testing та Black Box Testing, у яких тестувальнику відомо все про систему або відомі тільки основні інтерфейси системи відповідно.

Використовуються такі методи тестування:

1. Мануальне тестування інтерфейсу.
2. Тестування продуктивності веб-додатку.
3. Тестування функціональних точок системи.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Повний процес тестування проходить у такому порядку:

1. Тестування інтерфейсу при різних роздільних здатностях екрану.
2. Тестування на відповідність функціональним вимогам.
3. Тестування полей вводу на граничні та неможливі значення.
4. Тестування кількості запитів до API.
5. Тестування у різних браузерах: Chrome, Safari, Firefox.
6. Тестування зручності використання.
7. Тестування при максимальному навантаженні комп’ютера на оперативну пам’ять та процесор.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А.Дичка

"___" _____ 2019 р.

ПРОГРАМНА СИСТЕМА ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ ЗА ДОПОМОГОЮ АНАЛІЗУ СОЦІАЛЬНИХ МЕДІА АЛГОРИТМАМИ МАШИННОГО НАВЧАННЯ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ М.І. Гончар

2019

ЗМІСТ

1. ОПИС СТРУКТУРИ ПРОГРАМНОЇ СИСТЕМИ

Програмна система імплементована у вигляді веб-додатку, що складається із статичних та динамічних веб-сторінок, тобто таких, вміст яких залишається статичним та формується динамічно відповідно.

Веб-додаток є одномовним, використовується українська мова. На деяких позначеннях можливе використання англійської мови, наприклад, у прикладах коду або на графіках для позначення інтернаціональних позначень цін.

У системі існують такі статичні веб-сторінки:

1. Головна сторінка.
2. Сторінка входу до системи.
3. Сторінка реєстрації нового довіреного користувача.
4. Сторінка призначення системи.
5. Сторінка роботи алгоритмів системи.
6. Сторінка керівництва користувача.
7. Сторінка керівництва програміста.

У системі існують такі динамічні веб-сторінки:

1. Сторінка огляду криптовалют.
2. Сторінка перегляду криптовалютного ринку.
3. Сторінка перегляду останніх зібраних новин.
4. Сторінка оновлення системних даних.

Кожна веб-сторінка містить навігаційний елемент та елемент у нижньому кінці сторінки, що містять корисні посилання на усі необхідні сторінки у веб-додатку.

Якщо поточний користувач є автентифікований як довірена особа, то йому відкриваються посилання на вихід із системи та посилання на перегляд останніх зібраних даних.

Якщо поточний користувач є автентифікований як адміністратор, то йому відкриваються посилання на панель Django Admin, посилання на

перегляд останніх зібраних даних, посилання на оновлення системних даних, посилання на реєстрацію нових довірених осіб, посилання на вихід із системи.

2. ВМІСТ СТАТИЧНИХ ВЕБ-СТОРІНОК

Головна сторінка містить короткий опис системи та посилання на сторінки, що відтворюють основні результати роботи системи. На головній сторінці можна також перейти до сторінки входу до системи адміністратором або довіреною особою.

Сторінка входу до системи зображена на рис. 2.1. На цій сторінці користувачу пропонується ввести пару значень логін та пароль для входу до системи. При введенні невірних даних буде виведено повідомлення про помилку, що показано на рис. 2.1. При введені коректних даних користувача буде перенаправлено до головної сторінки системи.

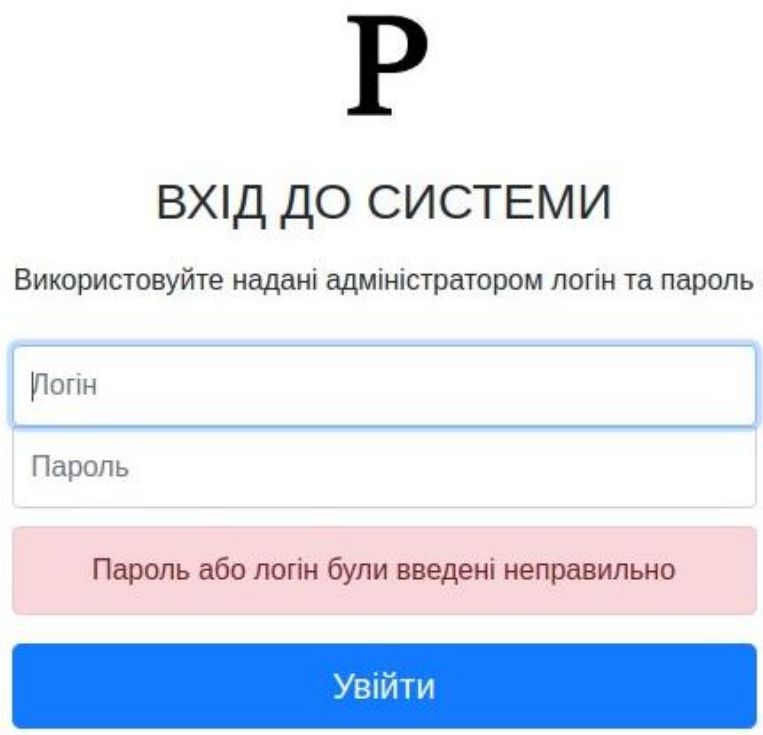


Рис. 2.1. Сторінка входу до системи

Адміністратор веб-додатку має право створювати довірених користувачів на сторінці «Реєстрація довіреної особи», знімок екрану якої показаний на рис 2.2. Доступ до цієї сторінки має лише адміністратор ресурсу. На цій сторінці адміністратору ресурсу пропонується ввести поля, що необхідні для створення нової довіреної особи: логін і пароль для входу, а також підтвердження паролю. При успішному створенні нового користувача під формою виводиться повідомлення результату із іменем нового користувача, що показано на рис. 2.2. При неуспішному створенні на місці повідомлення про успіх виводиться повідомлення про помилку. Поле логін користувача має обмеження на свій вміст: максимум 150 символів, можна використовувати лише латинські букви, цифри та деякі символи. Поле паролю має містити понад 8 символів, а також не має складатись лише із загальновживаних слів.

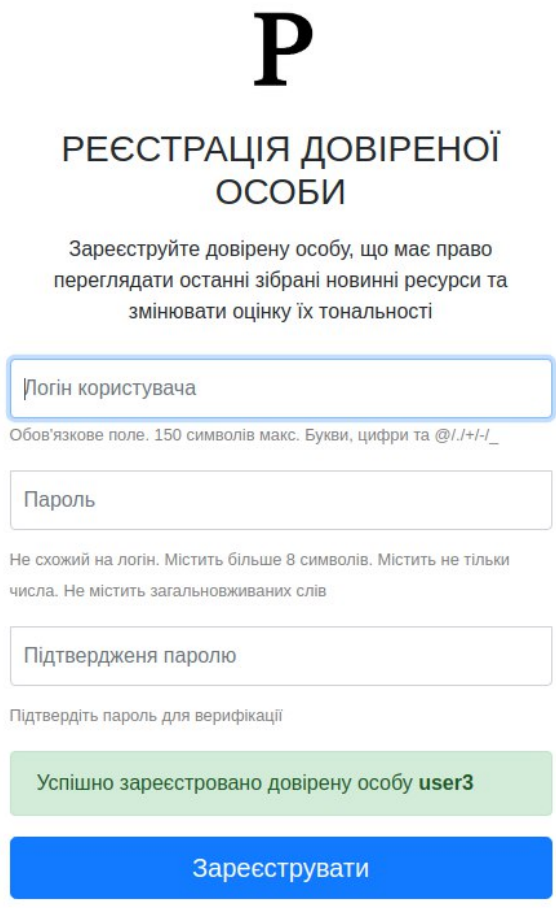


Рис. 2.2. Сторінка реєстрації довіреної особи

На сторінці керівництва користувача указана корисна інформація та поради для найефективнішого користування веб-додатком. На сторінці про призначення системи наведено інформацію про основну функціональність системи, а також зазначено розробника програмного забезпечення. На сторінці про роботу системи наведено інформацію про основні алгоритми для визначення тональності тексту та створення передбачень значень часового ряду, що використовуються у веб-додатку.

Веб-система надає вільний доступ до свого API стороннім розробникам. Правила використання такого програмного інтерфейсу наведені на сторінці опису API системи. Також наведено посилання на вихідний код системи та загальну інформацію про обмеження частоти запитів до функціональних точок системи. Для кожного запиту наведено його опис та приклад відповіді серверної частини програмного забезпечення.

3. ПРОЦЕДУРА АВТОРИЗАЦІЇ ДОВІРЕНОЇ ОСОБИ

Авторизація користувача відбувається на сторінці авторизації веб-додатку. Користувач має ввести поля «логін» та «пароль», які він отримав від адміністратора. Після введення правильних значень користувач має натиснути на кнопку «Увійти», після чого його буде перенаправлено на головну сторінку.

4. АДМІНІСТРУВАННЯ СИСТЕМИ

Дії щодо адміністрування системи заключаються в оновленні історичних та останніх текстових і фінансових даних. Такі дії може виконувати лише адміністратор, оскільки сторінка оновлення системних даних доступна лише користувачу із такою роллю.

Адміністратор ресурсу має можливість вручну змінювати системні дані: оновлювати історичні фінансові та текстові дані на сторінці «Оновлення системних даних», що показана на рис. 4.1. На цій сторінці можна оновити як за останній час (приблизно 1 тиждень), так і за будь-який часовий проміжок від 2016-01-01. При натисненні будь-якої кнопки на сторінці з'являється колесо, що крутиться, тим самим показуючи той факт, що система виконує оновлення даних. При успішному зборі даних на місці колеса з'являється повідомлення про успішно завантажені дані, а при неуспішному зборі даних – повідомлення про помилку.

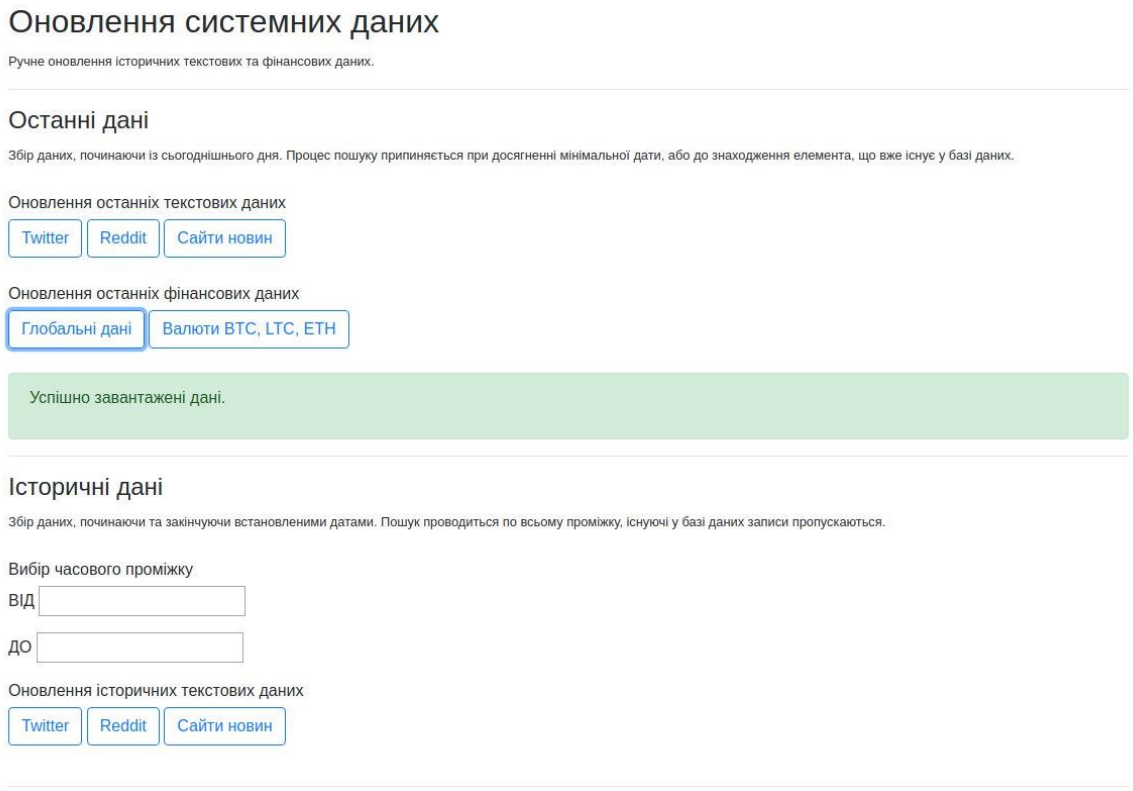


Рис. 4.1. Сторінка оновлення системних даних

5. ПЕРЕГЛЯД ПЕРЕДБАЧЕНЬ

У системі існують декілька видів передбачень, усі вони відтворені у системі на відповідних графіках.

Розглянемо передбачення узагальненого криптовалютного ринку. Таке передбачення зображене на сторінці «Ринок». Основний графік та дані, пов’язані із передбаченням, зображено на рис. 5.1. На сторінці показано розраховане передбачення капіталізації на наступну годину та розрахований прогноз тональності тексту на наступну годину. На графіку даних зображена інформація від 2018-01-01 по сьогоднішній день про загальну капіталізацію, об’єм торгівлі за 24 години, передбачення капіталізації, передбачення тональності текстів про ринок цифрових валют. Графік є інтерактивним: можна видаляти непотрібні графіки та передивлятись значення у деякій точці на графіку, навівши курсором миші на необхідну точку.

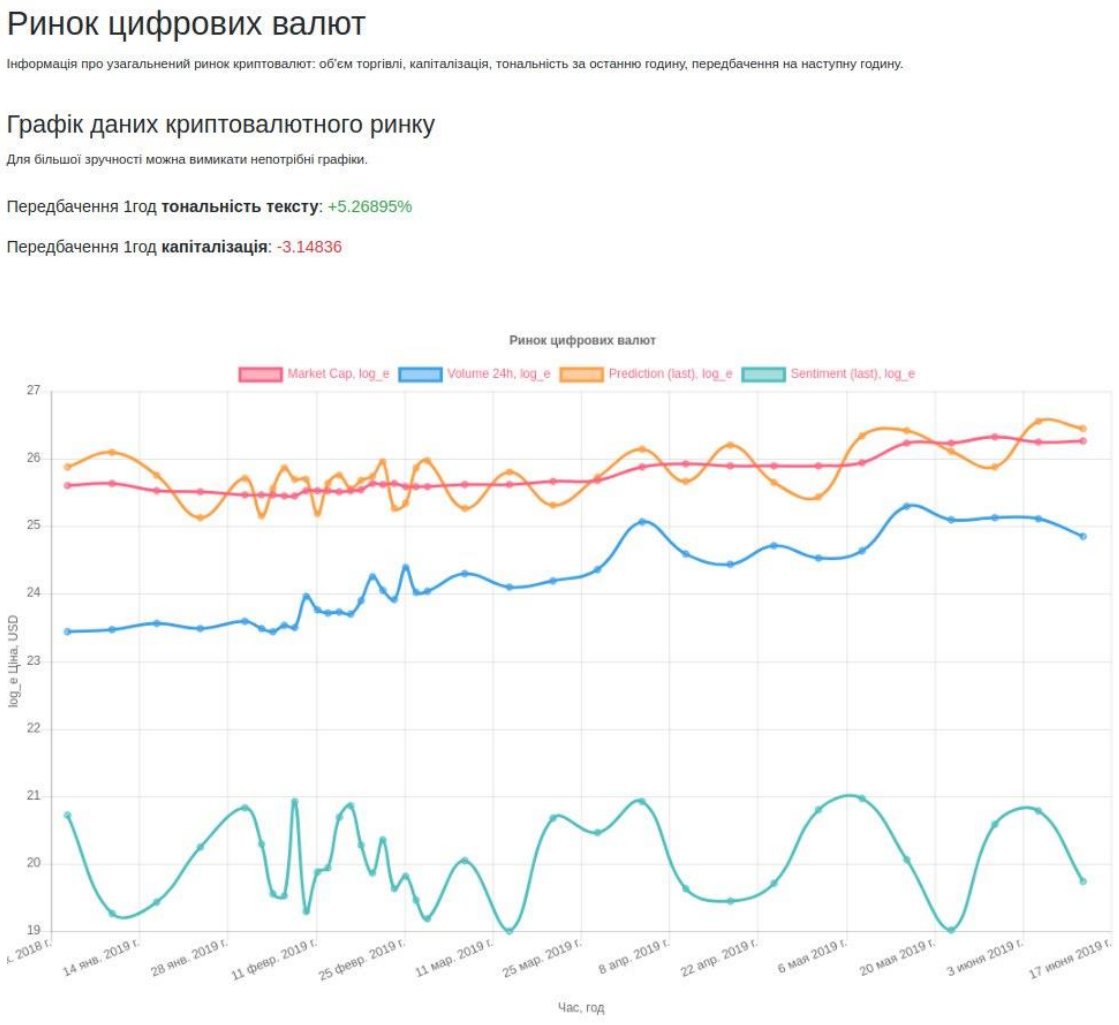


Рис. 5.1. Сторінка узагальненого ринку цифрових валют